

IN THE U.S. PATENT AND TRADEMARK OFFICE

Applicant(s): KAMITANI, Shingo

Application No.:

Group:

Filed: June 13, 2001

Examiner:

For: DATA-DRIVEN PROCESSOR HAVING MULTIPLE-PRECISION DATA
PROCESSING FUNCTION AND DATA PROCESSING METHOD THEREOF



L E T T E R

Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

June 13, 2001
0033-0731P

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55(a), the applicant hereby claims the right of priority based on the following application(s):

<u>Country</u>	<u>Application No.</u>	<u>Filed</u>
JAPAN	2000-178733	06/14/00

A certified copy of the above-noted application(s) is(are) attached hereto.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to deposit Account No. 02-2448 for any additional fees required under 37 C.F.R. 1.16 or under 37 C.F.R. 1.17; particularly, extension of time fees.

Respectfully submitted,

BIRCH, STEWART, KOLASCH & BIRCH, LLP

By: 

TERRELL C. BIRCH
Reg. No. 19,382
P. O. Box 747
Falls Church, Virginia 22040-0747

Attachment
(703) 205-8000
/rem

日本国特許庁
JAPAN PATENT OFFICE

6-13-01
BSKB
(703) 205-8501
0033-0731P
10F1

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2000年 6月14日

出願番号

Application Number:

特願2000-178733

出願人

Applicant(s):

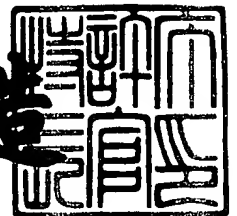
シャープ株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

2001年 4月27日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2001-3035730

【書類名】 特許願

【整理番号】 1000803

【提出日】 平成12年 6月14日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/00

【発明者】

 【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号 シャープ株式会社内

 【氏名】 紙谷 晋吾

【特許出願人】

 【識別番号】 000005049

 【住所又は居所】 大阪府大阪市阿倍野区長池町 2 2 番 2 2 号

 【氏名又は名称】 シャープ株式会社

【代理人】

 【識別番号】 100064746

 【弁理士】

 【氏名又は名称】 深見 久郎

【手数料の表示】

 【予納台帳番号】 008693

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 データ駆動型処理装置およびデータ駆動型処理装置におけるデータ処理方法

【特許請求の範囲】

【請求項 1】 k ビット長 (k は任意の正の整数) の n 個 (n は 2 以上の任意の整数) の領域を含むメモリを備えたデータ駆動型処理装置であって、

前記 n 個領域に格納されたメモリデータのそれぞれを演算処理する演算処理部をさらに備えて、

前記演算処理部は、

前記メモリデータと、与えられるデータパケットのデータフィールドに格納された前記 k ビット長のデータとを所定演算命令に従い所定演算処理して、前記所定演算処理の結果を、複数の前記 k ビット長のデータを分割して、前記分割により得られた前記複数の k ビット長のデータのそれぞれを前記データフィールドに格納した複数の前記データパケットを出力する所定演算手段と、

前記所定演算手段から出力された前記複数のデータパケットを入力して、入力した複数データパケットのそれぞれについて、

該データパケットの前記データフィールドの前記 k ビット長のデータを、前記メモリの所定アドレスに対応の前記領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、前記桁あふれしたデータを前記データフィールドに格納した前記データパケットを出力するデータ累積加算手段と、

前記桁あふれしたデータを前記データフィールドに格納した前記データパケットを入力して、該データパケットの前記データフィールドの前記桁あふれしたデータを、前記メモリにおける前記所定アドレスとは異なる上位の所定アドレスに対応の前記領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、前記桁あふれしたデータを前記データフィールドに格納した前記データパケットを出力するあふれデータ累積加算手段とを有して、

前記累積加算により前記所定アドレスの領域において桁あふれしたデータが

生じる間は、前記あふれデータ累積加算手段による前記桁あふれしたデータについての前記累積加算が繰返されることを特徴とする、データ駆動型処理装置。

【請求項 2】 前記演算処理部は、前記累積加算により前記所定アドレスの領域において桁あふれしたデータが生じているか否か判定する桁あふれ判定手段をさらに有して、

前記桁あふれ判定手段により桁あふれしたデータが生じていると判定されたことに応じて、前記あふれデータ累積加算手段により前記桁あふれしたデータについての前記累積加算が行なわれることを特徴とする、請求項 1 に記載のデータ駆動型処理装置。

【請求項 3】 2 個の m (m は $n * k \geq m$ を満たす任意の整数) ビット長の多倍精度データ同士を前記所定演算処理する場合に、

一方の前記多倍精度データを前記 k ビット長ごとに分割して得られたそれぞれのデータは、前記メモリの n 個の領域のそれぞれに前記メモリデータとして格納されて、

他方の前記多倍精度データが前記 k ビット長ごとに分割して得られたそれぞれのデータは、 n 個の前記データパケットのそれぞれの前記データフィールドに格納されて、 n 個の前記データパケットは前記所定演算手段に順次与えられることを特徴とする、請求項 1 または 2 に記載のデータ駆動型処理装置。

【請求項 4】 前記データパケットは、該データパケットを一意に識別するための世代番号が格納される世代フィールドをさらに有して、

所定アドレスは、前記データパケットの前記世代フィールドの内容に基づいて指定されることを特徴とする、請求項 1 ないし 3 のいずれかに記載のデータ駆動型処理装置。

【請求項 5】 前記データ累積加算手段および前記あふれデータ累積加算手段のそれぞれの、

与えられるデータパケットを入力して、該入力データパケット中の前記データフィールドの内容を、前記メモリの前記所定アドレスに対応の前記領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、前記桁あふれしたデータを前記入力データパケットの前記データフィール

ドに格納して、該入力データ packets を出力することを指示する演算命令に従い動作することを特徴とする、請求項 1 ないし 4 のいずれかに記載のデータ駆動型処理装置。

【請求項 6】 k ビット長 (k は任意の正の整数) の n 個 (n は 2 以上の任意の整数) の領域を含んで、前記 n 個の領域のそれぞれにメモリデータが格納されるメモリを備えたデータ駆動型処理装置におけるデータ処理方法であって、

前記メモリデータと、与えられるデータ packets のデータフィールドに格納された前記 k ビット長のデータとを所定演算命令に従い所定演算処理して、前記所定演算処理の結果を、複数の前記 k ビット長のデータを分割して、前記分割により得られた前記複数の k ビット長のデータのそれぞれを前記データフィールドに格納した複数の前記データ packets を出力する所定演算ステップと、

前記所定演算ステップから出力された前記複数のデータ packets を入力して、入力した複数データ packets のそれぞれについて、

該データ packets の前記データフィールドの前記 k ビット長のデータを、前記メモリの所定アドレスに対応の前記領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、前記桁あふれしたデータを前記データフィールドに格納した前記データ packets を出力するデータ累積加算ステップと、

前記桁あふれしたデータを前記データフィールドに格納した前記データ packets を入力して、該データ packets の前記データフィールドの前記桁あふれしたデータを、前記メモリにおける前記所定アドレスとは異なる上位の所定アドレスに対応の前記領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、前記桁あふれしたデータを前記データフィールドに格納した前記データ packets を出力するあふれデータ累積加算ステップとを有して、

前記累積加算により前記所定アドレスの領域において桁あふれしたデータが生じる間は、前記あふれデータ累積加算ステップによる前記桁あふれしたデータについての前記累積加算が繰返されることを特徴とする、データ駆動型処理装置におけるデータ処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明はデータ駆動型処理装置およびデータ駆動型処理装置におけるデータ処理方法に関し、特に、多倍精度形式のデータ（以下、多倍精度データという）についてデータ駆動型の演算を行なうデータ駆動型処理装置およびデータ駆動型処理装置におけるデータ処理方法に関する。

【0002】

【従来の技術および発明が解決しようとする課題】

大量データの高速処理が望まれる場合には、並列処理が有効である。並列処理向きアーキテクチャのうちでも、データ駆動型と呼ばれるものが特に注目される。

【0003】

データ駆動型情報処理システムでは、「ある処理に必要な入力データがすべて揃い、かつその処理に必要な演算装置などの資源が割当てられたときに処理が行なわれる」という規則に従って処理が並列に進行する。

【0004】

図13は、従来およびこの発明の実施の形態に適用されるデータ駆動型情報処理システムのブロック構成図である。図14は、従来のデータ駆動型処理装置の構成図である。図15（A）と（B）は従来およびこの発明の実施の形態に適用されるデータパケットのフィールド構成図である。図15（A）では、データ駆動型処理装置の入出力データパケットPAの基本構成が示される。図15（B）では、データ駆動型処理装置内部を流れるデータパケットPA1の基本構成が示される。

【0005】

図15（A）のデータパケットPAはプロセッサ番号PE（Processing Element）を格納するフィールド18、ノード番号Nを格納するフィールド19、世代番号Gを格納するフィールド20およびデータDを格納するフィールド21を含む。図15（B）のデータパケットPA1は、図15（A）と同様のフィールド

19～21と、命令コードCを格納するフィールド22とを含む。

【0006】

図13においてデータ駆動型情報処理システムは従来のデータ駆動型処理装置1（本実施の形態に適用されるデータ駆動型処理装置10）、複数のデータが予め格納されるデータメモリ3およびメモリインターフェース2を含む。データ駆動型処理装置1（10）はデータ伝送路4、5および9のそれぞれが接続される入力ポートIA、IBおよびIV、ならびにデータ伝送路6、7および8のそれぞれが接続される出力ポートOA、OBおよびOVを含む。

【0007】

データ駆動型処理装置1（10）はデータ伝送路4または5から入力ポートIAまたはIBを介して、データパケットPAが、時系列的に入力される。データ駆動型処理装置1（10）には所定の処理内容がプログラムとして予め記憶されており、そのプログラム内容に基づく処理が実行される。

【0008】

メモリインターフェース2はデータ駆動型処理装置1（10）の出力ポートOVから出力されてデータメモリ3に対するアクセス（データメモリ3の内容の参照／更新など）要求を、データ伝送路8を介して受理する。メモリインターフェース2は受理したアクセス要求に従ってメモリアクセス制御線SSLを介してデータメモリ3に対してアクセスを行なった後、その結果を、データ伝送路9および入力ポートIVを介してデータ駆動型処理装置1（10）に与える。

【0009】

データ駆動型処理装置1（10）は、入力したデータパケットPAに対する処理をして、処理が終了した後、出力ポートOAおよびデータ伝送路6または出力ポートOBおよびデータ伝送路7を介してデータパケットPAを出力する。

【0010】

図14には、従来のデータ駆動型処理装置1の構成が示される。図において、データ駆動型処理装置1は入出力制御部11、合流部12、データ駆動型の処理を行なうために発火制御部13、内蔵メモリ15が接続される演算部14およびプログラム記憶部16ならびに分岐部17を含む。

【0011】

ここで図15(A)と(B)を参照すると、プロセッサ番号PEは、複数のデータ駆動型処理装置1が接続されたシステムにおいて対応するデータパケットPAが処理されるべきデータ駆動型処理装置1を特定するための情報である。ノード番号Nは、プログラム記憶部16の内容をアクセスするためのアドレスとして用いられる。世代番号Gは、データ駆動型処理装置1に時系列に入力されるデータパケットを一意に識別するための識別子として用いられる。また世代番号Gはデータメモリ3が画像データメモリであった場合には、データメモリ3をアクセスするためのアドレスとしても用いられる。その際には、世代番号Gは上位ビットから順にフィールド番号F#、ライン番号L#およびピクセル番号P#を示す。

【0012】

動作において、図15(A)のデータパケットPAはデータ伝送路4、5を介してプロセッサ番号PEで指定されたデータ駆動型処理装置1に与えられると入出力制御部11において図15(B)のデータパケットPA1となる。つまり入出力制御部11は、入力したデータパケットPAのプロセッサ番号PEのフィールド18を破棄して、該入力データパケットPAのノード番号Nに基づいて、命令コードCと新たなノード番号Nとを得て、該入力データパケットPAのフィールド18と19にそれぞれ格納して、データパケットPA1を合流部12に出力する。したがって、入出力制御部11から合流部12に与えられたデータパケットPA1は図15(B)の構成を有する。なお、入出力制御部11では世代番号GとデータDは変化しない。

【0013】

合流部12は、入出力制御部11から与えられるデータパケットPA1、ならびに分岐部17から出力されるデータパケットPA1を順次入力して、発火制御部13に出力する。

【0014】

発火制御部13には、対となるデータパケットPA1を検出する(これを発火という)ための待合せメモリ731と定数データが1つ以上格納される定数デー

タメモリ 7 3 2 が含まれる。発火制御部 1 3 は、待合せメモリ 7 3 1 を利用して合流部 1 2 から与えられるデータパケット P A 1 について必要に応じて待合せを行なう。この結果、ノード番号 N および世代番号 G が一致する 2 つのデータパケット P A 1、すなわち対となる異なる 2 つのデータパケット P A 1 のうち一方のデータパケット P A 1 のフィールド 2 1 のデータ D を、他方のデータパケット P A 1 のフィールド 2 1 に追加して格納して、この他方のデータパケット P A 1 を演算部 1 4 に出力する。このとき一方のデータパケット P A 1 は消去される。ここでは、演算されるべき相手がデータパケット P A 1 ではなく定数データである場合には、発火制御部 1 3 での待合せは行なわれず、定数データが定数データメモリ 7 3 2 から読出されてデータパケット P A 1 のフィールド 2 1 に追加して格納されて、該データパケット P A 1 は演算部 1 4 に出力される。

【 0 0 1 5 】

演算部 1 4 は発火制御部 1 3 から与えられたデータパケット P A 1 を入力して、データパケット P A 1 の命令コード C を解読して、解読結果に基づいて、所定の処理を行なう。命令コード C がデータ D を含むデータパケット P A 1 の内容に対する演算命令を示す場合には該命令コード C に従いデータパケット P A 1 の内容について所定の演算処理が施されて、その結果は該データパケット P A 1 に格納されて、該データパケット P A 1 はプログラム記憶部 1 6 に出力される。また、このとき、データパケット P A 1 の命令コード C がメモリアクセス命令を指示している場合には内蔵メモリ 1 5 へのアクセス処理が行なわれて、アクセス結果を格納したデータパケット P A 1 はプログラム記憶部 1 6 に出力される。なお、演算部 1 4 に接続されるメモリはデータ駆動型処理装置 1 に内蔵されるメモリ 1 5 に限定されず該装置に外付けされるメモリであってもよい。

【 0 0 1 6 】

また演算部 1 4 は、命令コード C がデータメモリ 3 に対するアクセス命令を示す場合にはアクセス要求として該データパケット P A 1 を、データ伝送路 8 を介してメモリインターフェース 2 に与える。

【 0 0 1 7 】

メモリインターフェース 2 は、データ伝送路 8 を介して与えられたデータパケ

ットPA1を入力して、該入力データパケットPA1の内容に従って、メモリアクセス制御線SSLを介してデータメモリ3をアクセスする。そのアクセスの結果は該入力データパケットPA1のフィールド21にデータDとして格納されて、該データパケットPA1はデータ伝送路9を介して演算部14に与えられる。

【0018】

プログラム記憶部16は、複数の次位の命令コードCおよびノード番号Nからなるデータフロープログラムが格納されたプログラムメモリ161を有する。プログラム記憶部16は、演算部14から与えられたデータパケットPA1を入力し、該入力データパケットPA1のノード番号Nに基づくアドレス指定によって、次位のノード番号Nおよび次位の命令コードCをプログラムメモリ161から読出し、読出したノード番号Nおよび命令コードCを、該入力データパケットPA1のフィールド19および22のそれぞれに格納して、該入力データパケットPA1を分岐部17に出力する。

【0019】

分岐部17は、与えられたデータパケットPA1の命令コードCが該データ駆動型処理装置1内の演算部14で実行されるべきものか、外部のデータ駆動型処理装置1の演算部14で実行されるべきものを判別する。外部のデータ駆動型処理装置1の演算部14で実行されるべきと判別された場合にはデータパケットPA1が入出力制御部11に出力されて、入出力制御部11はデータパケットPA1を適切な出力ポートから該装置の外部に出力する。一方、該データ駆動型処理装置1内の演算部14で実行すべきと判別された場合は、データパケットPA1は合流部12に与えられる。

【0020】

このようにして、データパケットPA1がデータ駆動型処理装置1内を周回することにより、プログラムメモリ161に予め記憶されたデータフロープログラムに従う処理が進行する。

【0021】

データパケットはデータ駆動型処理装置1内においてはハンドシェイクによって非同期に転送される。プログラムメモリ161に格納されたデータフロープロ

グラムに従う処理は、データパケットがデータ駆動型処理装置 1 内を周回することによるパイプライン処理に従い並列に実行される。よって、データ駆動型処理方法によれば、データパケット単位での処理の並列性が高く装置内を周回するデータパケットのフローレートが処理性能の 1 つの尺度となる。

【 0 0 2 2 】

近年はこのようなデータ駆動型処理方法の特徴が、大量の演算を高速で行なうことが必要とされる画像処理あるいは映像信号処理へと応用される。画像や映像信号の性質上、これらに対応のデータのビット長は短い。したがって、画像処理あるいは映像信号処理においても短いビット長のデータが処理対象とされる。現在、図 1 5 (A) と (B) のデータ D のフィールド 2 1 は 1 2 b i t 長を有する。同様に、データメモリ 3 や内蔵メモリ 1 5 における 1 ワードも、 1 2 b i t 長を有する。

【 0 0 2 3 】

上述したような画像処理あるいは映像信号処理とは異なり処理対象とされるデータのビット長が非常に長い処理もある。このような処理としては、たとえば公開された鍵を用いた暗号化処理である公開鍵暗号化処理やそのための復号化処理がある。

【 0 0 2 4 】

ここで、上述の公開鍵暗号化処理について説明する。周囲には秘密にして、特定の相手にだけある文（データ）を伝えたいとき、その伝えたい文（データ）を平文と呼び、平文を相手に伝達するために暗号化処理を施したものを暗号文と呼ぶ。平文をある法則によって暗号文へ変換する（暗号化する）あるいは暗号文を平文へ変換する（復号化する）ためのパラメータを鍵と呼ぶ。公開鍵暗号化方式では、数学的な性質が利用されることにより暗号文や公開鍵が第三者にわかってもし受信者が互いに独自に持っている秘密の鍵が知られなければ暗号文を解読できない、または容易には解読できない仕組みとなっている。公開鍵暗号化方式の代表的なものとしては R S A (Rivest , Shamir, Adleman の略) や D H (Diffie Hellman の略) がある。以下、一例として D H の鍵交換について説明する。

【 0 0 2 5 】

鍵交換を行なう2人をAとBとする。AとBは、自分の秘密鍵 $S(A)$ および $S(B)$ のそれぞれを生成し、これを使って自分の公開鍵 $P(A)$ および $P(B)$ のそれぞれを、次の方法で作成する。なお、秘密鍵 $S(A)$ および $S(B)$ のそれぞれは1024bit長のデータである。公開鍵暗号化処理では秘密鍵は一般的に1024bit長を有する。

【0026】

公開鍵 $P(A) = G^{S(A)} \bmod P$ および公開鍵 $P(B) = G^{S(B)} \bmod P$ で求められる。ここで“ \wedge ”は冪乗算を示し“ \bmod ”は剰余算を示す。また、変数 G および P の値は定数として予め定められている。AとBは、お互いに生成した公開鍵を相手に送信し、各人が相手の公開鍵を受信すると、次のように共通鍵 C が作成される。つまり、Aは、 $C = P(B)^{S(A)} \bmod P$ により共通鍵 C を作成し、Bは $C = P(B)^{S(A)} \bmod P$ に従い共通鍵 C を作成する。

【0027】

2人が求めた共通鍵 C は全く同じ値となり、こうして秘密鍵を第三者に知られることなく送受信者で鍵の共有を図ることができる。なお $S(A)$ 、 $S(B)$ および P は1024bit長のデータであり、 $P(A)$ 、 $P(B)$ および C もまた1024bit長のデータである。

【0028】

上述した公開鍵の作成で使用される“ $X^Y \bmod Z$ ”という式に従う演算結果を求める際には、 X を定数とする乗算または2乗演算と、 Z を除数とする除算とが交互に繰返し行なわれる。またこの繰返し計算の中間結果を格納するために作業領域 U (2048bit)および V (2048bit)が準備される。“ $X^Y \bmod Z$ ”の演算のための処理フローが図16に示される。

【0029】

図16はノイマン型計算機において $X^Y \bmod Z$ の演算を実行するための処理フローチャートである。図16の処理フローを説明する。変数 X 、 Y および Z は、それぞれ102bit長で構成されている。これら変数の値は、計算機内の内部メモリに格納されており処理開始時に内部メモリから読出される。その後、

交互に中間演算とその結果格納が行なわれながら演算が進行する。なお、処理フローにおいて変数 $Y[k]$ は変数 Y の k ビット目の値を示す。

【 0 0 3 0 】

まず、ステップ $S1$ において初期設定がなされる。つまり作業領域 U の内容はリセットされて作業領域 V の内容には 1 が設定される。そして制御変数 k に 1023 がセットされる。つまり、制御変数 k が 1023 から 0 まで 1 ずつデクリメントされながら、以下の演算が繰返される。

【 0 0 3 1 】

ステップ $S2$ では、変数 $Y[k]$ が 1 であるか 0 であるかで処理は分岐する。もし、変数 $Y[k]$ が 1 であれば、ステップ $S3$ の処理に移行するが、 0 であれば後述のステップ $S6$ に移行する。

【 0 0 3 2 】

ステップ $S3$ では、 $V \times X$ の演算が行なわれ、その結果が作業領域 U に格納される。次のステップ $S4$ では $U \% Z$ に従う演算がなされて、つまり（作業領域 U に格納された値 $\div Z$ ）の剰余が求められて、その剰余値が作業領域 V に格納される。次のステップ $S5$ では、制御変数 k が 0 であるか否かが判定される。 0 でなければステップ $S6$ において、作業領域 V の値が 2 乗されて、その結果が作業領域 U に格納される。そして、次のステップ $S7$ では $U \% Z$ に従う演算がなされて、すなわち（作業領域 U に格納された値 $\div Z$ ）の剰余が求められて、その剰余値が作業領域 V に格納される。次のステップ $S8$ においては、制御変数 k の値が 1 デクリメントされる。以降 $S2 \sim S8$ の処理が、ステップ $S5$ において $k = 0$ と判定されるまで繰返される。その結果、作業領域 V に格納されている値が、“ $X \wedge Y \bmod Z$ ” の演算結果値となる。

【 0 0 3 3 】

このように公開鍵暗号化処理および復号化処理に代表されるように多倍精度データを処理する要求が生じているが、従来のデータ駆動型処理装置 1 によって多倍精度データを処理する方式はまだ確立されていない。詳述すると、公開鍵暗号化処理で必要とされる演算のビット長は 1024 bit 程度であり、データ駆動型処理装置 1 によって、そのようなビット長を有した演算器、データパケットお

よびメモリの1ワードを構成することは、データ駆動型処理装置1をLSI（集積回路の略）を用いて実現する場合の回路実装面積およびバス幅などの物理的な制約上非常に困難である。

【0034】

それゆえにこの発明の目的は、多倍精度データを効率よく処理することのできるデータ駆動型処理装置およびデータ駆動型処理装置におけるデータ処理方法を提供することである。

【0035】

【課題を解決するための手段】

この発明のある局面に係るデータ駆動型処理装置は、 k ビット長（ k は任意の正の整数）の n 個（ n は2以上の任意の整数）の領域を含むメモリを備えたデータ駆動型処理装置であって、さらに、 n 個領域に格納されたメモリデータのそれぞれを演算処理する演算処理部を備える。

【0036】

演算処理部は、所定演算手段と、データ累積加算手段と、あふれデータ累積加算手段とを有する。

【0037】

所定演算手段は、メモリデータと、与えられるデータパケットのデータフィールドに格納された k ビット長のデータとを所定演算命令に従い所定演算処理して、所定演算処理の結果を、複数の k ビット長のデータを分割して、分割により得られた複数の k ビット長のデータのそれぞれをデータフィールドに格納した複数のデータパケットを出力する。

【0038】

所定演算手段から出力された複数のデータパケットを入力して、入力した複数データパケットのそれぞれについて、データ累積加算手段と、あふれデータ累積加算手段とが適用される。

【0039】

データ累積加算手段は該データパケットのデータフィールドの k ビット長のデータを、メモリの所定アドレスに対応の領域のメモリデータに累積加算して、桁

あふれしたデータを除く累積加算結果を該領域に格納して、桁あふれしたデータをデータフィールドに格納したデータパケットを出力する。

【0040】

あふれデータ累積加算手段は、桁あふれしたデータをデータフィールドに格納したデータパケットを入力して、該データパケットのデータフィールドの桁あふれしたデータを、メモリにおける所定アドレスとは異なる上位の所定アドレスに対応の領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、桁あふれしたデータをデータフィールドに格納したデータパケットを出力する。累積加算により所定アドレスの領域において桁あふれしたデータが生じる間は、あふれデータ累積加算手段による桁あふれしたデータについての累積加算は繰返される。

【0041】

上述のデータ駆動型処理装置によれば、多倍精度データについて所定演算処理が行なわれる際には、多倍精度データは k ビット長単位に分割された複数の単精度データとして、そしてメモリの k ビット長の n 個の領域の集合は該多倍精度データとして扱われる。それゆえに、データ駆動型処理装置において多倍精度データ専用の累算器を特別に用意しなくても通常のメモリの領域で多倍精度データの累算器の役割を果たすことができるから、多倍精度データについて所定演算処理が行なわれる際でも、データ駆動型処理装置の小型化は阻害されない。

【0042】

また多倍精度データの所定演算では、多倍精度データは k ビット長の単精度データ単位の互いに独立な演算要素に分割することによってデータについての演算をすべて同時並列に実行できるから、データ駆動型処理装置の並列処理能力を最大限に発揮することができる。

【0043】

また、データ駆動型処理装置では、複数の演算の同時並列処理能力に加えてパイプライン並列処理という特徴を有しているから、多倍精度データの所定演算処理において、メモリを用いたデータの累算時に桁あふれが連続的に発生しても連続発生する桁あふれによる処理の遅延量は他の単精度データの処理によって抑制

される。つまりオーバーフローの発生度合いは多倍精度データ演算処理全体にかかる所要時間（レスポンス）にほとんど影響しない。

【0044】

さらに多倍精度データのデータ長に関して論理的な制約がない。データ駆動型処理装置の通常のメモリの領域やデータパケットが利用されるから、物理的に資源の許す限りどんなに長いビット長の多倍精度データであっても所定演算処理できる。しかも、ビット長が異なる多種類の多倍精度データを同時に扱うこともできる。

【0045】

上述の所定演算処理は、加算、減算、除算および乗算のいずれであってもよい。

【0046】

上述のデータ駆動型処理装置において、演算処理部は、累積加算により所定アドレスの領域において桁あふれしたデータが生じているか否か判定する桁あふれ判定手段をさらに有して、桁あふれ判定手段により桁あふれしたデータが生じていると判定されたことに応じて、あふれデータ累積加算手段により桁あふれしたデータについての累積加算が行なわれる。

【0047】

上述のデータ駆動型処理装置によれば、桁あふれ判定手段を設けることにより、桁あふれしたデータが生じている間は、あふれデータ累積加算手段により桁あふれしたデータについての累積加算が行なわれる。それゆえに、桁あふれしたデータを確実に繰上げながら所定演算を実行することができる。

【0048】

上述のデータ駆動型処理装置において、2個の m （ m は $n * k \geq m$ を満たす任意の整数）ビット長の多倍精度データ同士を所定演算処理する場合には、以下の特徴を有する。つまり、一方の多倍精度データを k ビット長ごとに分割して得られたそれぞれのデータは、メモリの n 個の領域のそれぞれにメモリデータとして格納されて、他方の多倍精度データが k ビット長ごとに分割して得られたそれぞれのデータは、 n 個のデータパケットのそれぞれのデータフィールドに格納され

て、 n 個のデータパケットは所定演算手段に順次与えられる。

【0049】

上述のデータ駆動型処理装置によれば、2個の m (m は $n * k \geq m$ を満たす任意の整数) ビット長の多倍精度データ同士を所定演算処理することができる。

【0050】

上述のデータ駆動型処理装置では、データパケットは、該データパケットを一意に識別するための世代番号が格納される世代フィールドをさらに有して、所定アドレスは、データパケットの世代フィールドの内容に基づいて指定される。

【0051】

上述のデータ駆動型処理装置によれば、累算器として機能するメモリの領域の所定アドレスは、データパケットを一意に識別するための世代番号が格納される世代フィールドの内容に基づいて指定される。したがって、多倍精度データを分割して得られた複数の k ビット長の単精度データのそれぞれについて対応する所定アドレスが一意に指定されることになって、メモリにおける累算を確実に行うことができる。

【0052】

上述のデータ駆動型処理装置において、データ累積加算手段およびあふれデータ累積加算手段のそれぞれは、以下の特徴を有する。つまり、与えられるデータパケットを入力して、該入力データパケット中のデータフィールドの内容を、メモリの所定アドレスに対応の領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、桁あふれしたデータを入力データパケットのデータフィールドに格納して、該入力データパケットを出力することを指示する演算命令に従い動作する。

【0053】

上述のデータ駆動型処理装置によれば、データ累積加算手段およびあふれデータ累積加算手段のそれぞれにおいては、同一種類の演算命令が実行される。したがって、多倍精度データを所定演算する場合であっても、実行に必要とされる演算命令の種類数を少なくできて、プログラムのメンテナンスおよびデバッグは容易である。

【 0 0 5 4 】

この発明の他の局面に係るデータ駆動型処理装置におけるデータ処理方法は、 k ビット長（ k は任意の正の整数）の n 個（ n は2以上の任意の整数）の領域を含んで、 n 個の領域のそれぞれにメモリデータが格納されるメモリを備えたデータ駆動型処理装置におけるデータ処理方法であって、所定演算ステップと、データ累積加算ステップと、あふれデータ累積加算ステップとを有する。

【 0 0 5 5 】

所定演算ステップでは、メモリデータと、与えられるデータパケットのデータフィールドに格納された k ビット長のデータとが所定演算命令に従い所定演算処理されて、所定演算処理の結果は、複数の k ビット長のデータに分割されて、分割により得られた複数の k ビット長のデータのそれぞれがデータフィールドに格納された複数のデータパケットが出力される。

【 0 0 5 6 】

データ累積加算ステップとあふれデータ累積加算ステップとは、所定演算ステップにより出力された複数のデータパケットを入力して、入力した複数データパケットのそれぞれについて適用される。

【 0 0 5 7 】

データ累積加算ステップでは、該データパケットのデータフィールドの k ビット長のデータは、メモリの所定アドレスに対応の領域のメモリデータに累積加算されて、桁あふれしたデータを除く累積加算結果は該領域に格納されて、桁あふれしたデータがデータフィールドに格納されたデータパケットが出力される。

【 0 0 5 8 】

あふれデータ累積加算ステップでは、桁あふれしたデータをデータフィールドに格納したデータパケットが入力されると、該データパケットのデータフィールドの桁あふれしたデータは、メモリにおける所定アドレスとは異なる上位の所定アドレスに対応の領域のメモリデータに累積加算されて、桁あふれしたデータを除く累積加算結果は該領域に格納されて、桁あふれしたデータがデータフィールドに格納されたデータパケットが出力される。上述の累積加算により所定アドレスの領域において桁あふれしたデータが生じる間は、あふれデータ累積加算ステ

ップによる桁あふれしたデータについての累積加算は繰返される。

【 0 0 5 9 】

上述のデータ駆動型処理装置におけるデータ処理方法によれば、多倍精度データについて所定演算処理が行なわれる際には、多倍精度データはkビット長単位に分割された複数の単精度データとして、そしてメモリのkビット長のn個の領域の集合は該多倍精度データとして扱われる。それゆえに、データ駆動型処理装置において多倍精度データ専用の累算器を特別に用意しなくても通常のメモリの領域で多倍精度データの累算器の役割を果たすことができるから、多倍精度データについて所定演算処理が行なわれる際でも、データ駆動型処理装置の小型化は阻害されない。

【 0 0 6 0 】

また多倍精度データの所定演算では、多倍精度データはkビット長の単精度データ単位の互いに独立な演算要素に分割することによってデータについての演算をすべて同時並列に実行できるから、データ駆動型処理装置のデータ処理における並列処理能力を最大限に発揮することができる。

【 0 0 6 1 】

また、データ駆動型処理装置のデータ処理方法は、複数の演算の同時並列処理能力に加えてパイプライン並列処理という特徴を有しているから、多倍精度データの所定演算処理において、メモリを用いたデータの累算時に桁あふれが連続的に発生しても連続発生する桁あふれによる処理の遅延量は他の単精度データの処理によって抑制される。つまりオーバーフローの発生度合いは多倍精度データ演算処理全体にかかる所要時間（レスポンス）にほとんど影響しない。

【 0 0 6 2 】

さらに多倍精度データのデータ長に関して論理的な制約がない。データ駆動型処理装置の通常のメモリの領域やデータパケットが利用されるから、物理的に資源の許す限りどんなに長いビット長の多倍精度データであっても所定演算処理できる。しかも、ビット長が異なる多種類の多倍精度データを同時に扱うこともできる。

【 0 0 6 3 】

上述の所定演算処理は、加算、減算、除算および乗算のいずれであってもよい。

【0064】

上述のデータ処理方法において、演算処理ステップは、累積加算により所定アドレスの領域において桁あふれしたデータが生じているか否か判定する桁あふれ判定ステップをさらに有して、桁あふれ判定ステップにより桁あふれしたデータが生じていると判定されたことに応じて、あふれデータ累積加算ステップにより桁あふれしたデータについての累積加算が行なわれる。

【0065】

上述のデータ処理方法によれば、桁あふれ判定ステップを設けることにより、桁あふれしたデータが生じている間は、あふれデータ累積加算ステップにより桁あふれしたデータについての累積加算が行なわれる。それゆえに、桁あふれしたデータを確実に繰上げながら所定演算を実行することができる。

【0066】

上述のデータ駆動型処理方法において、2個の m (m は $n * k \geq m$ を満たす任意の整数) ビット長の多倍精度データ同士を所定演算処理する場合には、以下の特徴を有する。つまり、一方の多倍精度データを k ビット長ごとに分割して得られたそれぞれのデータは、メモリの n 個の領域のそれぞれにメモリデータとして格納されて、他方の多倍精度データが k ビット長ごとに分割して得られたそれぞれのデータは、 n 個のデータパケットのそれぞれのデータフィールドに格納されて、 n 個のデータパケットは所定演算ステップに順次与えられる。

【0067】

上述のデータ駆動型処理方法によれば、2個の m (m は $n * k \geq m$ を満たす任意の整数) ビット長の多倍精度データ同士を所定演算処理することができる。

【0068】

上述のデータ駆動型処理方法では、データパケットは、該データパケットを一意に識別するための世代番号が格納される世代フィールドをさらに有して、所定アドレスは、データパケットの世代フィールドの内容に基づいて指定される。

【0069】

上述のデータ駆動型処理方法によれば、累算器として機能するメモリの領域の所定アドレスは、データパケットを一意に識別するための世代番号が格納される世代フィールドの内容に基づいて指定される。したがって、多倍精度データを分割して得られた複数のkビット長の単精度データのそれぞれについて対応する所定アドレスが一意に指定されることになって、メモリにおける累算を確実に行うことができる。

【0070】

上述のデータ駆動型処理方法において、データ累積加算ステップおよびあふれデータ累積加算ステップのそれぞれは、以下の特徴を有する。つまり、与えられるデータパケットを入力して、該入力データパケット中のデータフィールドの内容を、メモリの所定アドレスに対応の領域のメモリデータに累積加算して、桁あふれしたデータを除く累積加算結果を該領域に格納して、桁あふれしたデータを入力データパケットのデータフィールドに格納して、該入力データパケットを出力することを指示する演算命令に従い動作する。

【0071】

上述のデータ駆動型処理方法によれば、データ累積加算ステップおよびあふれデータ累積加算ステップのそれぞれにおいては、同一種類の演算命令が実行される。したがって、多倍精度データを所定演算する場合であっても、実行に必要とされる演算命令の種類数を少なくできて、プログラムのメンテナンスおよびデバッグは容易である。

【0072】

【発明の実施の形態】

以下、この発明の実施の形態について説明する。

【0073】

まず、本実施の形態の特徴について説明する。

本実施の形態では、データ駆動型処理装置において多倍精度データの演算を実現するために、LSIとして実現できる現実的な範囲での短いビット長を有したデータやメモリワードを使用して多倍精度データの演算処理を実現する。なお、ここでメモリワードとは、データ駆動型処理装置における1回のメモリアクセス

命令の実行によりアクセスできるメモリのデータ量を示す。

【0074】

図1は、この発明の実施の形態に係る演算部141の構成を、入出力されるデータパケットとともに示す図である。図2は、図1のswitch回路65の構成を入出力されるデータとともに示す図である。図3はこの発明の実施の形態に係るデータ駆動型処理装置10のブロック図である。図4(A)と(B)は、本実施の形態による多倍精度データの分割を説明する図である。

【0075】

図3のデータ駆動型処理装置10は、図13のシステムにおいて従来のデータ駆動型処理装置1に代替して設けられる。図3のデータ駆動型処理装置10と図14の従来のデータ駆動型処理装置1と比較し異なる点は、データ駆動型処理装置10がデータ駆動型処理装置1の演算部14および内蔵メモリ15に代替して演算部141および累算メモリ151を有した内蔵メモリ150を備える点にある。データ駆動型処理装置10の他の部分はデータ駆動型処理装置1と同様であり説明を省略する。

【0076】

図1を参照して、演算部141は従来の演算部14の構成に加えてメモリ累算命令ACCMCを実行するACCMC回路68を追加実装している。演算部141はDEMUX回路64、分岐命令switchを実行するswitch回路65、命令inc_genを実行するinc_gen回路66、乗算命令MUL32を実行するMUL32回路67、ACCMC回路68、他の種類の演算を行なう演算回路69～71、およびMUX（マルチプレクサの略）回路72を含む。

【0077】

switch回路65、inc_gen回路66およびMUL32回路67は従来から提供される回路である。命令switchと命令inc_genについては、後述する。

【0078】

演算部141にはデータパケットPA1(IN)が入力される。データパケットPA1(IN)は命令コードC、ノード番号N、世代番号G、およびフィールド

ド 2 1 のデータ D として左データ L D および右データ R D が格納されている。左データ L D および右データ R D は、命令コード C が 2 項演算命令などであった場合に、発火制御部 1 3 における待合せによって得られた 2 つのオペランドデータである。ただし、命令コード C が 2 項演算命令であっても、演算対象となる 2 つのオペランドデータ的一方が定数である場合には発火制御部 1 3 において定数データメモリ 7 3 2 から読出された定数データが格納される。

【 0 0 7 9 】

データパケット P A 1 (I N) が演算部 1 4 1 に入力されると、入力データパケット P A 1 の命令コード C は、D E M U X 回路 6 4 と M U X 回路 7 2 に与えられる。D E M U X 回路 6 4 は、与えられる命令コード C に基づいて演算回路 6 5 ～ 7 1 のうちのいずれか 1 つを選択して、選択した回路に該入力データパケット P A 1 (I N) を与える。演算回路 7 1 および A C C M C 回路 6 8 は必要に応じて内蔵メモリ 1 5 0 をアクセスする。各演算回路は、与えられるデータパケット P A 1 (I N) の内容を、対応の命令コード C に基づいて演算し、その演算結果をフィールド 2 1 に格納したデータパケット P A 1 (I N) を M U X 回路 7 2 に与える。

【 0 0 8 0 】

M U X 回路 7 2 は与えられる命令コード C に基づいて、演算回路 6 5 ～ 7 1 のいずれか 1 つの出力を選択して入力する。そして、入力したデータパケットは P A 1 (O U T) として出力されて、プログラム記憶部 1 6 に与えられる。

【 0 0 8 1 】

データパケット P A 1 (O U T) は命令コード C、ノード番号 N、世代番号 G およびフィールド 2 1 においてデータ D および真偽フラグ F L を格納する。

【 0 0 8 2 】

真偽フラグ F L は、命令 s w i t c h を含む分岐命令の実行結果によって出力される 1 ビットのフラグデータである。命令 s w i t c h による判定結果が「真」のときは真偽フラグ F L に 1 が設定され、「偽」のときには 0 が設定される。分岐命令以外の命令では、真偽フラグ F L には「真」を表わす 1 が常に出力される。この真偽フラグ F L によって次位の命令コード C とノード番号 N とがプログラ

ム記憶部 1 6 のプログラムメモリ 1 6 1 から読出される。つまり、命令 `switch` の次に実行される命令として真偽フラグ `FL` の値によって、たとえば、命令 `end_judge` および命令 `inc_gen` のいずれか一方がプログラムメモリ 1 6 1 から選択的に読出される。

【 0 0 8 3 】

図 2 には図 1 の `switch` 回路 6 5 の構成例が示される。`switch` 回路 6 5 はロジック回路 `LU` を含む。`switch` 回路 6 5 はデータパケット `PA1 (IN)` のフィールド 2 1 のビットデータを参照した分岐機能とメモリ累算命令 `ACCMC` の出力値は 0 であるかどうかの判定機能を有する。`switch` 回路 6 5 では、データパケット `PA1 (IN)` のフィールド 2 1 のデータ `D` の下位 1 ビットの値 (1 または 0) がロジック回路 `LU` に与えられる。ロジック回路 `LU` は、与えられる下位 1 ビットのデータが 0 のときには真偽フラグ `FL` として 1 を出力し、1 のときは真偽フラグ `FL` として 0 を出力する。したがって、真偽フラグ `FL` が 1 のときは、メモリ累算命令 `ACCMC` の実行結果として出力値 0 が得られたことが判定される。また入力データパケット `PA1 (IN)` のフィールド 2 1 のデータは出力データパケット `PA1 (OUT)` のフィールド 2 1 のデータ `D` として格納される。

【 0 0 8 4 】

本実施の形態では演算対象となる多倍精度データを、データ駆動型処理装置 1 0 をフローするデータパケットのフィールド 2 1 に格納可能なビット長を有した複数のデータに分割する。仮に、1 0 2 4 `bit` の多倍精度データは 3 2 `bit` の 3 2 個のデータに分割されると想定する。この分割の様子が図 4 (A) と (B) に示される。図 4 (A) の 1 0 2 4 `bit` の多倍精度データ `A` は、図 4 (B) の 3 2 `bit` のデータ `A [0] ~ A [31]` に分割される。データ `A [0] ~ A [31]` のそれぞれは、対応するそれぞれのデータパケット `PA1` のフィールド 2 1 に格納されるとともに、データ `A [j]` (ただし $j = 0, 1, 2, \dots, 31$) が多倍精度データ `A` におけるいずれの桁 (ビット) の位置に対応するかを示す j の値に相当する情報がフィールド 2 0 の一部として格納される。ただし、フィールド 2 0 への正確な格納方法は実際の演算の形態に依存する。

【0085】

このようにして1024bitの多倍精度データAは、32個のデータパケットPA1の集合として表現される。図4(B)の32個のデータパケットPA1それぞれについては、フィールド21に格納されるデータA[0]～A[31]のそれぞれのみが示されて、他のフィールドのデータは省略されている。

【0086】

図5(A)と(B)は、累算メモリ151の領域における多倍精度データの格納例を説明する図である。内蔵メモリ150の累算メモリ151の領域は、多倍精度データの演算の中間結果や最終結果を格納するためのメモリとしてアクセスされる。累算メモリ151の領域は、多倍精度データを演算することが可能なビット長を有したメモリワード単位に分割される。仮に、2048bitの多倍精度データの演算を想定すると、この多倍精度データは、32bit長の64個のデータに分割されて累算メモリ151の領域に格納される。この様子が図5(A)と(B)に示される。

【0087】

図5(A)の2048bitの多倍精度データCは、図5(B)のような32bitのデータC[0]～C[63]へと分割された後に、これらデータは累算メモリ151のアドレス0～63のメモリワードのそれぞれに格納される。データC[h] (h=0、1、2、…、63)が多倍精度データCのうちのいずれの桁(ビット)に相当するかを示すhの値に相当する情報は、アドレス0～63に相当する。ここでいうアドレスは、あるオフセットを基準(アドレス0)として相対値として与えられる。このようにして2048bitの多倍精度データCは累算メモリ151では64個のメモリワードの列として表現される。

【0088】

また、64個のメモリワードの列は2048bitの記憶領域を有する累算器として機能させることが所望されるが、これらのメモリワード列は、64個の全く独立した32bit長のメモリワードの集合にすぎない。しかしながら、メモリを用いた累算は、データパケットPA1のフィールド21の32bit長のデータDとメモリの32bit長の1メモリワードとを対象に行なわれるので

、前述した64個のメモリワードを多倍精度データの累算器として機能させるには、32bit単位の累算時に生じる桁あふれ、すなわちオーバーフローまたはアンダーフローを、次位のアドレスのメモリワードに反映させなければならない。

【0089】

言換えると、「データD (32bit) + 所定メモリワードに格納されたデータ (32bit)」の結果値が32bitで表現できる値を超えてオーバーフローした場合、所定メモリワードには結果値の下位32bitだけが格納されて、所定メモリワードの次位のアドレスのメモリワードの内容にオーバーフローした1が加算されなければならない。逆に「データD (32bit) - 所定メモリワードに格納されたデータ (32bit)」の結果値が負の値になってアンダーフローした場合、所定メモリワードには結果値の下位32bitだけが格納され、所定メモリワードの次位のアドレスのメモリワードの内容からアンダーフローした1を引かなければならない。そのために、データ駆動型処理装置10の演算部141は、図1に示されるようにオーバーフローおよびアンダーフローを処理するためのメモリアクセスに関する命令を実行するための機能を、従来の機能に追加して備えている。

【0090】

図6は、メモリワードでオーバーフローが生じた例を説明する図である。図6ではデータパケットPA1については、フィールド21の内容のみが示されて、他のフィールドの図示は省略されている。仮に、データパケットPA1のデータDを、ある指定アドレスのメモリワードの内容に累算し、その結果オーバーフローが生じた場合の例が図6に示される。

【0091】

図6において、メモリ累算命令ACCMCを用いずに、「データパケットPA1のフィールド21のデータA[x]はメモリワードC[x]へ累算して、その累算結果をメモリワードC[63]～C[0]へと反映させる」という命令セットを用いて処理することを想定すると処理の流れは次のようになる。

【0092】

(A 1) のデータパケット P A 1 の世代番号 G から累算メモリ 1 5 1 のアドレスを算出する。たとえば、アドレス x が算出されたとする。

【 0 0 9 3 】

(A 2) メモリワード C [x] の値を累算メモリ 1 5 1 から読出す。

(A 3) 上述の (A 2) の結果とデータ A [x] を加算し加算結果 D とする (D → C [x] , A [x]) 。

【 0 0 9 4 】

(A 4) 上述の (A 3) の加算結果の下位 3 2 b i t のデータを累算メモリ 1 5 1 のアドレス x へ書込む。

【 0 0 9 5 】

(A 5) 上述の (A 3) の加算結果が 3 2 b i t で表現できる値を超えていないか (オーバーフローしていないか) を判定し、オーバーフローしていなければ終了し、オーバーフローしていれば次の処理に移行する。

【 0 0 9 6 】

(B 1) 次のメモリアドレス x + 1 を算出する。

(B 2) メモリワード C [x + 1] を累算メモリ 1 5 1 から読出す。

【 0 0 9 7 】

(B 3) 上述の (B 2) の読出値に 1 を加算して、加算結果を D とする (D → C [x + 1] + 1) 。

【 0 0 9 8 】

(B 4) 上述の (B 3) の加算結果の下位 3 2 b i t のデータを累算メモリ 1 5 1 のアドレス x + 1 へ書込む。

【 0 0 9 9 】

(B 5) 上述の (B 3) の加算結果が 3 2 b i t で表現できる値を超えていないか (オーバーフローしていないか) 判定し、オーバーフローしていなければ処理を終了し、オーバーフローしていれば x + 1 に 1 を加えて x + 2 として、上述の (B 1) ~ (B 5) の処理を繰返す。このオーバーフローは、 C [x + 1] = 0 x f f f f である限り発生する。

【 0 1 0 0 】

このような不定回数のオーバーフローが連続して発生する可能性を考慮すると、1回の実行で不定回数のメモリアクセスを指示するような命令を処理するためのハードウェアを従来の演算部14に実装することは、困難であった。なぜならば、上述の処理は加算結果がオーバーフローしなくなるまで繰返されて、繰返しの回数は、連続するオーバーフローの回数に依存しているため、この命令実行に要する時間は毎回異なるから、命令実行に要する時間が固定でない。また、データ駆動型処理装置は周回パイプライン構造になっており、データパケットPA1がこのパイプライン内をフローする場合には、全体においてデータパケットPA1の処理が滞りなく進行して、前段のパイプラインが空き状態にならないとデータパケットPA1は前段へは進めない。したがって、この命令で長い実行時間がかかると、該命令の実行はパイプラインの輻輳を引き起こすかもしれない。この輻輳を回避するのにはパイプラインの経路の分岐や緩衝バッファを追加するなどの特別な工夫が必要となる。

【0101】

また演算部のパイプラインにおいて、上述のような累算命令と他の命令とに基づいて、データパケットPA1の経路を分岐させたとしても、パイプラインの経路を、すなわち内蔵メモリ15の一部を累算器として用いることを想定しているから、上述のように内蔵メモリ15に関する不定回数のメモリアクセス実行中は他の命令によるメモリアクセスが妨げられるから、全体としての処理速度は低下する。

【0102】

そこで、本実施の形態では、上述したパイプラインにおける輻輳および処理速度低下というデメリットを回避するために、次のようになされる。つまり、オーバーフローが生じた場合には、累算メモリ151内の上位のメモリワードにオーバーフローした値(1)を累算するのではなくて、データパケットPA1のフィールド21にデータDとして格納して該データパケットPA1を出力するというメモリ累算命令ACCMCが提案される。

【0103】

図7は、本実施の形態に係るメモリ累算命令ACCMCを説明する図である。

図7では、データパケットPA1は、フィールド21に格納されたデータDの値のみが示されて、他のフィールドの内容の図示は省略されている。図7においては処理OP1とOP2が実行されている。まず、処理OP1においては、演算部141に命令コードCとして“ACCMC”を、データDとしてA[x]を、世代番号Gとしてxを格納したデータパケットPA1が与えられると、メモリ累算命令ACCMCに基づいて、累算メモリ151のアドレスxのメモリワードC[x]の内容が読み出されて、その値にデータパケットPA1のデータA[x]が累算されて、オーバーフローを含まない結果はメモリワードC[x]に書き込まれる。この累算で生じたオーバーフローの値(+1)を用いて、データパケットPA1のフィールド21の内容はA[x+1]に書き換えられて、該データパケットは演算部141から出力される。

【0104】

該データパケットPA1は該装置のパイプラインを1回以上周回した後に、演算部141に与えられると、処理OP2において、同様にして該データパケット中のメモリ累算命令ACCMCに従い累算メモリ151の次のアドレスx+1のメモリワードC[x+1]の内容が読み出されて、読み出された内容と該データパケットのデータA[x+1] (A[x+1]=1) とが累算されて、累算結果は、累算メモリ151のアドレスx+1のメモリワードC[x+1]に書き込まれる。これにより、累算と、累算により生じたオーバーフロー処理とを実行することができる。

【0105】

上述したように、連続してオーバーフローが発生しても、1つの累算命令実行当たり、累算メモリ151に関して読出と書込をそれぞれ1回行なうだけでよい。また、上述したように連続的にオーバーフローが発生して複数回の累算命令が連続的に実行されることになったとしても、データ駆動型処理装置におけるパイプライン処理による処理の並列性の特徴により、演算に関する遅延時間を抑制することができる。

【0106】

次に、上述した本実施の形態の特徴に基づいた具体的な動作を以下に説明する

【 0 1 0 7 】

ここで 1 0 2 4 b i t の多倍精度データ A と 1 0 2 4 b i t の多倍精度データ B とを乗算して 2 0 4 8 b i t の多倍精度データ C を得る演算処理を例示して説明する。

【 0 1 0 8 】

図 3 のデータ駆動型処理装置 1 0 のパイプラインを周回する図 1 5 (B) のデータパケット P A 1 は 3 2 b i t のフィールド 2 1 を有し、累算メモリ 1 5 1 では 1 メモリワード当り 3 2 b i t で構成されるものとする。

【 0 1 0 9 】

初期状態として多倍精度データ A と B は累算メモリ 1 5 1 に格納される。累算メモリ 1 5 1 では、多倍精度データ A はメモリワード A [0] ~ A [3 1] の 3 2 ワードに格納され、多倍精度データ B はメモリワード B [0] ~ B [3 1] の 3 2 ワードに格納される。乗算が実行されるときには、メモリワード A [i] とメモリワード B [j] (i , j = 0 , 1 , 2 , 3 … , 3 1) が累算メモリ 1 5 1 から必要に応じて読出されて演算に使用される。そして、乗算結果である 2 0 4 8 b i t の多倍精度データ C は、累算メモリ 1 5 1 のメモリワード C [0] ~ C [6 3] の 6 4 ワードで構成される。演算においては中間結果がメモリワード C [0] ~ C [6 3] の 6 4 ワードにその都度累算される。したがって、累算処理終了時点でのメモリワード C [0] ~ C [6 3] の 6 4 ワードにより最終の乗算結果が示される。

【 0 1 1 0 】

上述した $A * B$ の乗算処理を説明する。まず、乗算命令 MUL 3 2 が定義される。乗算命令 MUL 3 2 は $A * B$ についての各部分積データ (6 4 b i t) を求め、該部分積データを上位 3 2 b i t のデータと下位 3 2 b i t のデータに分けて、分けられたそれぞれのデータを各データパケットのフィールド 2 1 に格納して出力することを指示する。

【 0 1 1 1 】

また、ここでは上述したメモリ累算命令 ACCMC が定義される。演算部 1 4

1では乗算命令MUL 32の実行後、メモリ累算命令ACCMCが実行される。メモリ累算命令ACCMCが実行されることにより、データパケットPA1の世代番号Gに基づいて累算メモリ151の所定アドレスが算出されて、該所定アドレスによりアドレス指定される累算メモリ151のメモリワードから読出された値に、該データパケットPA1のフィールド21のデータDを累算する。そしてこの累算によりオーバーフローおよびアンダーフローした値を除く累算結果を該所定アドレスに対応する累算メモリ151のメモリワードに書込むとともに、オーバーフローおよびアンダーフローのいずれかが発生すればその値(+1または-1)を該データパケットPA1のフィールド21のデータDに代替して格納して、該データパケットPA1を出力する。

【0112】

次に上述した $A * B$ の乗算処理を、上述した乗算命令MUL 32およびメモリ累算命令ACCMCを用いて説明する。図8には本実施の形態による $A * B$ の乗算式が示されている。図示されるように多倍精度データA(1024bit)、多倍精度データB(1024bit)の乗算は、32bit同士の乗算結果(以下、部分積という)の総和であって図8の式(1)のように示される。図8の式(1)において、 $\ll (32 * 1)$ は左へ32ビットシフトとすることを示す。また $\ll (32 * 3)$ は左へ 32×3 ビットシフトすることを示す。

【0113】

図8の式(1)の各部分積 $A[i] * B[j]$ は64bit長を有する。これを、32bit長のフィールド21を有するデータパケットPA12つで表現する。この部分積は、演算部141のMUL 32回路67において乗算命令MUL 32が実行されることにより求められる。乗算命令MUL 32が実行されると、部分積 $A[i] * B[j]$ が求められて、該部分積は上位32bitのデータと下位32bitのデータに分けられて、各データをデータDとしてフィールド21に格納したデータパケットがそれぞれ出力される。

【0114】

図9は、本実施の形態に係る乗算命令MUL 32を用いた処理フローチャートである。図9では、32ビットのデータX($A[i]$)をデータDとして格納し

たデータパケットPA1と32ビットのデータY ($B[j]$) をデータDとして格納したデータパケットPA1とが発火制御部13にて発火して、命令コードCとして“MUL32”を、データDとしてデータX ($A[i]$) とY ($B[j]$) とが格納されたデータパケットPA(IN)が演算部141に与えられる。ただし、データパケットPA(IN)ではフィールド21および22の内容のみが示されて、他のフィールドの図示は省略される。

【0115】

そして、演算部141のMUL32回路67において、与えられたデータパケットPA(IN)が入力されて、入力データパケットPA(IN)のフィールド22に格納されていた乗算命令MUL32が実行される(ステップS1)。実行により得られた乗算結果の値Z ($A[i] * B[j]$) は上位の32bitのデータZu ($A[i] * B[j] \text{ __upper}$) と下位の32bitのデータZl ($A[i] * B[j] \text{ __lower}$) に分割されて、分割により得られた2つのデータは2つのデータパケットPA(OUT)のフィールド21にデータDとしてそれぞれ格納されて、2つのデータパケットPA(OUT)は演算部141から出力される(ステップS2)。ただし、データパケットPA(OUT)ではフィールド21の内容のみが示されて、他のフィールドの図示は省略される。

【0116】

データパケットPA1が部分積 $A[i] * B[j]$ に関するデータパケットの場合には、世代番号Gのフィールド20には、前述したように、添字iとjが格納される。具体的には、世代番号Gのフィールド20には2つの領域が含まれて、一方の領域には“i”が、もう一方の領域には“j”が格納されることによって部分積 $A[i] * B[j]$ の識別が行なわれる。そして、世代番号Gのフィールド20の2つの領域から部分積 $A[i] * B[j]$ を累算するためのメモリワードのアドレスを求めることができる。つまり、データ $A[i] * B[j] \text{ __lower}$ については、これら2つの領域の値を加えるだけで、累算に用いられるメモリワードのアドレス($i + j$)が得られ、データ $A[i] * B[j] \text{ __upper}$ については、これら2つの領域の値の加算値にさらに1を加えることにより累算に用いられるメモリワードのアドレス($i + j + 1$)が得られる。

【0117】

図10は、本実施の形態に係る部分積毎の処理内容（乗算と累算）を示す処理フローチャートである。なお、図10では、累算により、桁あふれは生じていないと想定している。図9の乗算命令MUL32が実行された後、図9のデータパケットPA（OUT）は、パイプラインを周回することにより、プログラム記憶部16において対応の命令コードCが累算命令ACCMCに更新されて、再度、演算部141に与えられる。

【0118】

演算部141のACCMC回路68に、与えられる2つのデータパケットPA（IN）は、データDとしてデータA[i] * B[j] _upperとデータA[i] * B[j] #lowerとをそれぞれ格納しており、命令コードCとして“ACCMC”を格納しており、世代番号Gとしてiとjが格納されている。演算部141のACCMC回路68は、2つのデータパケットPA（IN）を入力して、データA[i] * B[j] _upperを格納したデータパケットPA（IN）の世代番号Gに基づいて、累算メモリ151のアドレス（i + j + 1）のメモリワードC[i + j + 1]の内容を読み出して（ステップS1a）、読み出された値に該データパケットPA（IN）に格納されているデータA[i] * B[j] _upperを累算して、その結果値を累算メモリ151のアドレス（i + j + 1）のメモリワードC[i + j + 1]に書き込む（ステップS2a）。

【0119】

また、データA[i] * B[j] #lowerを格納したデータパケットPA（IN）の世代番号Gに基づいて、累算メモリ151のアドレス（i + j）のメモリワードC[i + j]の内容を読み出して（ステップS1b）、読み出された値に該データパケットPA（IN）に格納されているデータA[i] * B[j] _lowerを累算されて、その結果値を累算メモリ151のアドレス（i + j）のメモリワードC[i + j]に書き込む（S2b）。

【0120】

図10で示されるように、部分積を求める対象となっているデータA[j]、データB[j]の添字iとjを変化させていくことで、累算メモリ151で指定

されるメモリワードのアドレスも同様に変更されて、32ビット単位でシフト演算が行なわれていることになる。このことは図8の式(1)の部分積の和を表わす中で、32bitずつシフトすることに相当する。

【0121】

このように、累算メモリ151を用いて、求めた各部分積が累算されることで最終的に乗算結果C(2048bit)が求められることになる。

【0122】

図11は、本実施の形態に係るメモリ累算命令ACCMCを説明するためのフローチャートである。なお、図においては、データパケットPA(IN)およびPA(OUT)については、フィールド21と20のみが示されて他のフィールドの図示は省略されている。メモリ累算命令ACCMCが実行される際に、まずフィールド21に32bitのデータX(部分積 $A[i] * B[j]_{\text{lower}}$ または部分積 $A[i] * B[j]_{\text{upper}}$)とフィールド20の世代番号Gを格納したデータパケットPA(IN)が演算部141に与えられる。データパケットPA(IN)の世代番号Gのフィールド20は、前述したように部分積 $A[i] * B[j]$ を特定するためのiを格納する領域36とjを格納する領域37とを含む。演算部141のACCMC回路68は与えられるデータパケットPA(IN)を入力して、該入力データパケットPA(IN)のフィールド22のメモリ累算命令ACCMCに従って、フィールド20の内容を用いて $i + j + k$ が算出される(ステップS1)。なお、kは累算メモリ151の領域中のメモリワードC[0]を決めるための32bitの値(オフセット値)であって、定数である。

【0123】

そして、累算メモリ151のアドレス $i + j + k$ のメモリワードC[i+j]に格納されているデータMが読出されて(ステップS2)、読み出されたデータM(32bit)を左入力データとし、データパケットPA(IN)のフィールド21のデータXを右入力データとして加算が行なわれて、33bitの加算結果Zが得られると(ステップS3)、加算結果Zは下位32bitのデータと上位1ビットのデータとに分割されて(ステップS4)、下位32bitのデータ

は累算メモリ151のアドレス $i+j+k$ のメモリワード $C[i+j]$ に書込まれて(ステップS5)、上位1bitのデータはデータパケットPA(IN)のフィールド21にデータXに代替して格納されて該データパケットPA(IN)はデータパケットPA(OUT)として演算部141から出力される。

【0124】

データパケットPA(OUT)にデータDとして格納された加算結果の上位1bitのデータは加算結果によるオーバーフローの有無を表わし、この値が“1”であればオーバーフローが発生したことを、“0”であればオーバーフローが発生していないことを示す。そして、このデータパケットPA(OUT)のフィールド20には、データパケットPA(IN)のフィールド20の値 i および j のいずれかが1つ増加されて、 $i+j+k+1$ が格納される。これは、アンダーフローが生じた場合でも同様にして行なうことができる。

【0125】

ここで、仮にオーバーフローのデータパケットPA1が出力された場合、これを累算メモリ151へ反映させるためにオーバーフローした値をデータDとして格納したデータパケットを、パイプラインを周回させて、再度、演算部141のACCMC回路68に与えて、該データパケットの内容についてメモリ累算命令ACCMCを、同様にして実行する。この実行により、さらにオーバーフローが発生すれば累算メモリ151のさらに1つ上のアドレスに対してメモリ累算命令ACCMCが実行される。このような処理がオーバーフローが生じなくなるまで継続される。

【0126】

上述した乗算命令MUL32に従う乗算とメモリ累算命令ACCMCに従う累算処理を図8の式(1)のすべての部分積について実行すると、多倍精度データAとBの乗算終了は、すべての部分積に対するメモリ累算命令ACCMCの出力として“0”が得られたか、つまりオーバーフローが生じなくなったかどうかで判定できる。これは、図8の式(1)の部分積の総数(1024個)に対応してメモリ累算命令ACCMCの出力値“0”をデータDとして格納したデータパケットPA1が2048個($=1024 \times 2$)だけカウントされたことにより判定

される。

【0127】

図12は、本実施の形態による多倍精度データ同士の乗算に関する全体の処理を示すフローチャートである。図の処理フローは、`init_load`モジュール42、`init_stream`モジュール43、モジュール44、`next_stream`モジュール45、`inc_gen`モジュール46と49、`ACCM`モジュール47、`switch`モジュール48および`end_judge`モジュール50を含む。

【0128】

`init_load`モジュール42では、入力データA[0]～A[31]と入力データB[0]～B[31]を累算メモリ151に格納する。次の`init_stream`モジュール43では、世代番号の領域37の値(j)を0として、累算メモリ151のメモリワードA[0]～A[31]およびB[j]の計33個が累算メモリ151から読出される。次のモジュール44はMUL32回路67に相当して、乗算命令MUL32を前述のように実行する。MUL32命令は図9に示された命令である。

【0129】

次の`next_stream`モジュール45では、入力データパケットPA1のフィールド21の領域37の値(j)に1を加えて、累算メモリ151のメモリワードA[0]～A[31]とB[j]の計33個を累算メモリ151から読出す。上述したモジュール44と45の処理を、j=0～31まで順に繰返すことで、すべての部分積が求められる。これらの部分積の値の上位32bitのデータと下位32bitのデータとは、2つのデータパケットPA1のそれぞれのフィールド21に格納されて、これら2つのデータパケットPA1は`inc_gen`モジュール46に出力される。

【0130】

次に`inc_gen`モジュール46は`inc_gen`回路66に相当して、上位32bitのデータを格納しているデータパケットPA1の領域37の値(j)に1を加える。一方下位32bitのデータを格納しているデータパケットP

A 1 の領域 3 7 の値はそのままである。これは、前述したように累算メモリ 1 5 1 のメモリワードのアドレスに対応させるためになされる。

【 0 1 3 1 】

次にメモリ累算命令 ACCMC のモジュール 4 7 は、ACCMC 回路 6 8 に相当して、図 1 1 に示される処理を実行する。なお、図 1 1 の定数 k にはメモリワード C [0] のアドレスが格納される。

【 0 1 3 2 】

次の switch モジュール 4 8 は、switch 回路 6 5 に相当して、モジュール 4 7 から出力されたデータパケット PA 1 を順次入力して、オーバーフローの値を格納したデータパケット PA 1 とそうでないパケット PA 1 とを識別して、オーバーフローの値を格納したデータパケット PA 1 は inc_gen モジュール 4 9 に出力して、そうでないデータパケット PA 1 は end_judge モジュール 5 0 に出力する。この出力の分岐は次のようになされる。つまり、図 2 のロジック回路 LU が、与えられる下位 1 ビットのデータが 0 のときには真偽フラグ FL として 1 を出力し、1 のときは真偽フラグ FL として 0 を出力するので、真偽フラグ FL が 1 のときは、メモリ累算命令 ACCMC の実行結果として出力値 0 が得られたことがわかる。そして、switch モジュール 4 8 から出力された真偽フラグ FL の値に基づくプログラムメモリ 1 6 1 からの次位の命令コード C の読み出しは、命令 inc_gen および命令 end_judge のいずれか一方の選択的な読出しとなるから、上述した出力の分岐が実現される。

【 0 1 3 3 】

inc_gen モジュール 4 9 は、inc_gen 回路 6 6 に相当してモジュール 4 6 と同等の機能が行なわれる。end_judge モジュール 5 0 ではオーバーフローを示さない値 0 を持つデータパケット PA 1 の入力のカウントされて、カウント結果が 2 0 4 8 になったとき、演算処理の終了信号 END が出力される。このときの累算メモリ 1 5 1 内のメモリワード C [0] ~ C [6 3] で示される多倍精度データ C の値が、多倍精度データ A * B の最終の乗算結果を示す。

【 0 1 3 4 】

ここで、上述した実施の形態から得られる効果について説明する。

まず多倍精度データ A および B を複数の単精度データ A [i] および B [j] に分割し、累算メモリ 1 5 1 の固定長のメモリワードの集合を多倍精度データとして扱うことによって、データ駆動型処理装置 1 0 において多倍精度データ専用の累算器を特別に容易しなくても通常のメモリワードで多倍精度データの累算器の役割を果たすことができる。

【 0 1 3 5 】

また多倍精度データの演算を単精度データ単位の互いに独立な演算要素に分割することによってデータについての演算をすべて同時並列に実行できるから、データ駆動型処理装置 1 0 の並列処理能力を最大限に発揮することができる。データ駆動型処理方式は、複数の演算の同時並列処理能力に加えてパイプライン並列処理という特徴を有しているから、多倍精度データの乗算処理において、累算メモリ 1 5 1 を用いたデータの累算時にオーバーフローが連続的に発生しても連続発生するオーバーフローによる処理の遅延量は他の単精度データの処理によって抑制される。つまりオーバーフローの発生度合いは多倍精度データ演算処理全体にかかる所要時間（レスポンス）にほとんど影響しない。

【 0 1 3 6 】

さらに多倍精度データのデータ長に関して論理的な制約がない。データ駆動型処理装置 1 0 の通常のメモリワードやデータパケット P A (P A 1) が利用されるから物理的に資源の許す限りどんなに長いビット長のデータであっても演算処理できる。しかも、ビット長が異なる多種類の多倍精度データを同時に扱うこともできる。

【 0 1 3 7 】

上述の実施の形態では多倍精度データの乗算の例を示したが加算／減算／除算のいずれについても同様の処理方式を採用することができる。減算と除算の場合は図 1 1 の累算メモリ 1 5 1 の内容への累算とオーバーフロー出力命令の代わりに、累算メモリ 1 5 1 の内容からの減算とアンダーフロー出力命令を導入すればよい。

【 0 1 3 8 】

また、図12のデータフローグラフよりメモリ累算命令ACCMCを用いた多倍精度データの演算処理では累算メモリ151の32bitのデータを1回加算する際には、メモリ累算命令ACCMC、分岐命令switchおよび命令inc_genの3命令が実行される。したがって、この加算実行時に、n回のオーバーフローが発生すると、 $3 * n + 1$ 命令分の実行時間がかかる。

【0139】

そこで、メモリ累算命令ACCMCを用いずに前述したステップ(A1)～(B5)を実現する命令セットを実装する方が、累算メモリ151を用いた1回の加算処理を1命令時間で実行できてよいようにも見える。ところが図8の式(1)のような演算を実行する場合には、パイプライン型並列処理能力を十分に利用して、式(1)の複数の部分積が同時並列に計算される。この場合には、図3の周回パイプラインにおいて同時に多くのデータパケットPA1がフローする。したがって前述したようなパイプラインの輻輳および処理速度低下というデメリットが生じて、データ駆動型処理装置におけるパイプライン型並列処理能力を十分に発揮できなくなる。

【0140】

一方、メモリ累算命令ACCMCを用いて同様な処理を行う場合には、たとえ累算メモリ151への加算1回について $3 * n + 1$ 命令分の実行時間がかかったとしても、多倍精度データの演算処理全体はパイプライン型並列処理によって実効的な処理性能は飛躍的に向上する。

【0141】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【0142】

【発明の効果】

この発明に係るデータ駆動型処理装置およびデータ処理方法によれば、多倍精度データについて所定演算処理が行なわれる際には、多倍精度データはkビット

長単位に分割された複数の単精度データとして、そしてメモリのkビット長のn個の領域の集合は該多倍精度データとして扱われる。それゆえに、データ駆動型処理装置において多倍精度データ専用の累算器を特別に用意しなくても通常のメモリの領域で多倍精度データの累算器の役割を果たすことができるから、多倍精度データについて所定演算処理が行なわれる際でも、データ駆動方処理装置の小型化は阻害されない。

【0143】

また多倍精度データの所定演算では、多倍精度データはkビット長の単精度データ単位の互いに独立な演算要素に分割することによってデータについての演算をすべて同時並列に実行できるから、データ駆動型処理装置の並列処理能力を最大限に発揮することができる。

【0144】

また、データ駆動型処理装置では、複数の演算の同時並列処理能力に加えてパイプライン並列処理という特徴を有しているから、多倍精度データの所定演算処理において、メモリを用いたデータの累算時に桁あふれが連続的に発生しても連続発生する桁あふれによる処理の遅延量は他の単精度データの処理によって抑制される。つまりオーバーフローの発生度合いは多倍精度データ演算処理全体にかかる所要時間（レスポンス）にほとんど影響しない。

【0145】

さらに多倍精度データのデータ長に関して論理的な制約がない。データ駆動型処理装置の通常のメモリの領域やデータパケットが利用されるから、物理的に資源の許す限りどんなに長いビット長の多倍精度データであっても所定演算処理できる。しかも、ビット長が異なる多種類の多倍精度データを同時に扱うこともできる。

【図面の簡単な説明】

【図1】 この発明の実施の形態に係る演算部141の構成を、入出力されるデータパケットとともに示す図である。

【図2】 図1のswitch回路65の構成を入出力されるデータとともに示す図である。

【図 3】 この発明の実施の形態に係るデータ駆動型処理装置 1 0 のブロック図である。

【図 4】 (A) と (B) は、本実施の形態による多倍精度データの分割を説明する図である。

【図 5】 (A) と (B) は、累算メモリ 1 5 1 の領域における多倍精度データの格納例を説明する図である。

【図 6】 メモリワードでオーバーフローが生じた例を説明する図である。

【図 7】 本実施の形態に係るメモリ累算命令 ACCMC を説明する図である。

【図 8】 本実施の形態による $A * B$ の乗算式を示す図である。

【図 9】 本実施の形態に係る乗算命令 MUL 3 2 を用いた処理フローチャートである。

【図 1 0】 本実施の形態に係る部分積毎の処理内容（乗算と累算）を示す処理フローチャートである。

【図 1 1】 本実施の形態に係るメモリ累算命令 ACCMC を説明するためのフローチャートである。

【図 1 2】 本実施の形態による多倍精度データ同士の乗算に関する全体の処理を示すフローチャートである。

【図 1 3】 従来およびこの発明の実施の形態に適用されるデータ駆動型情報処理システムのブロック構成図である。

【図 1 4】 従来のデータ駆動型処理装置の構成図である。

【図 1 5】 (A) と (B) は従来およびこの発明の実施の形態に適用されるデータパケットのフィールド構成図である。

【図 1 6】 ノイマン型計算機において $X \wedge Y \bmod Z$ の演算を実行するための処理フローチャートである。

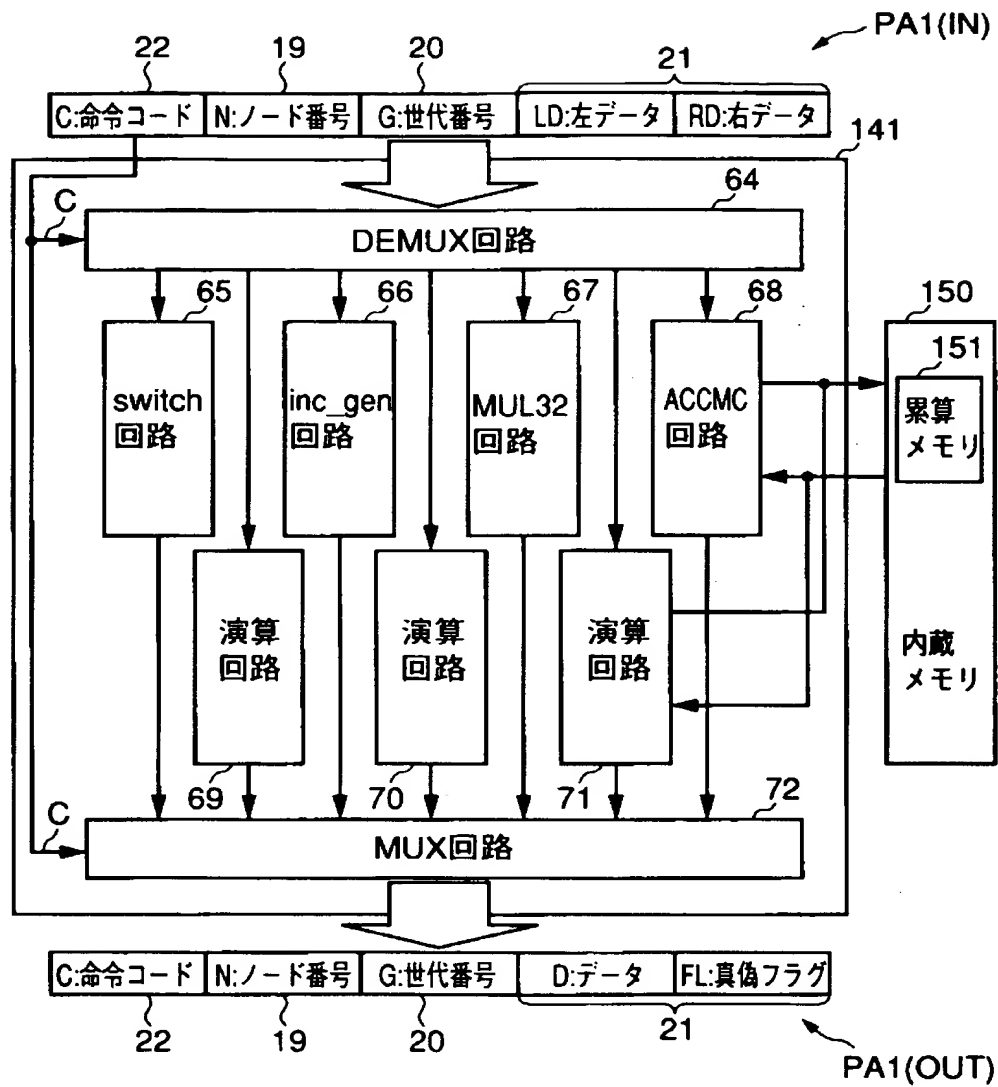
【符号の説明】

1、1 0 データ駆動型処理装置、1 1 入出力制御部、1 2 合流部、1 3 発火制御部、1 4、1 4 1 演算部、1 5、1 5 0 内蔵メモリ、1 5 1 累算メモリ、1 6 プログラム記憶部、1 7 分岐部、PA、PA 1、PA 1 (I

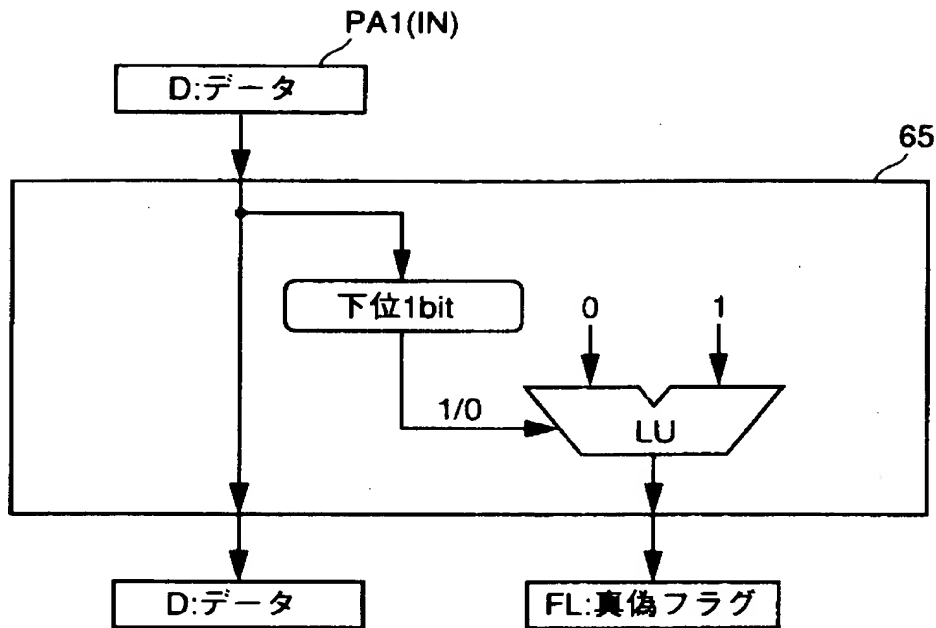
N)、PA1 (OUT) データパケット。

【書類名】 図面

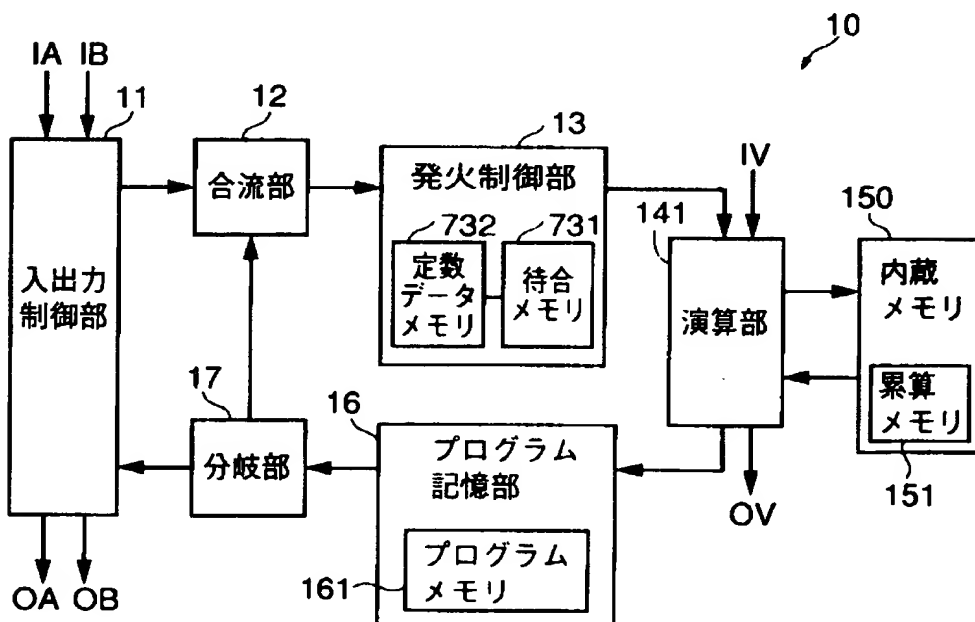
【図 1】



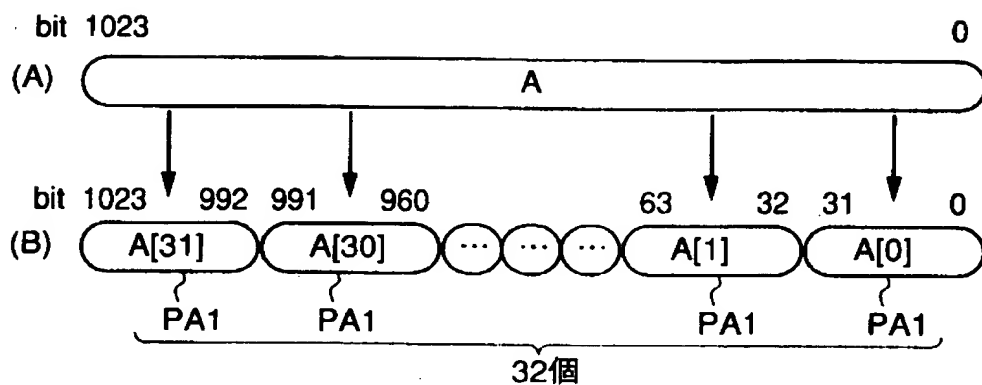
【図 2】



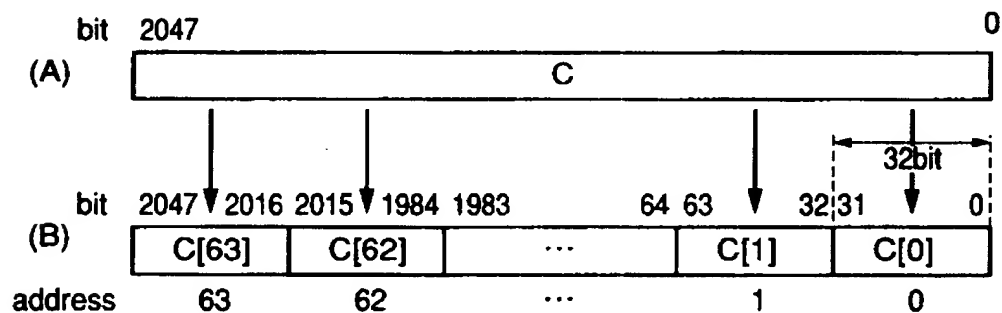
【図 3】



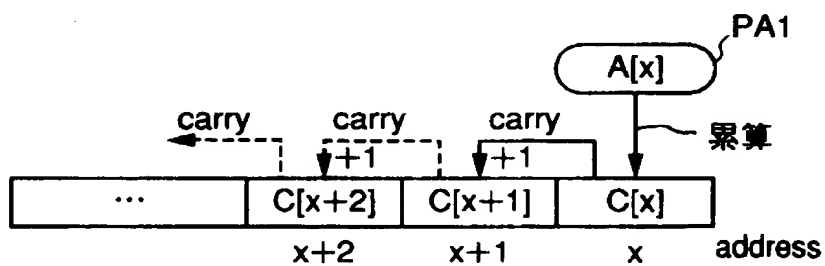
【図 4】



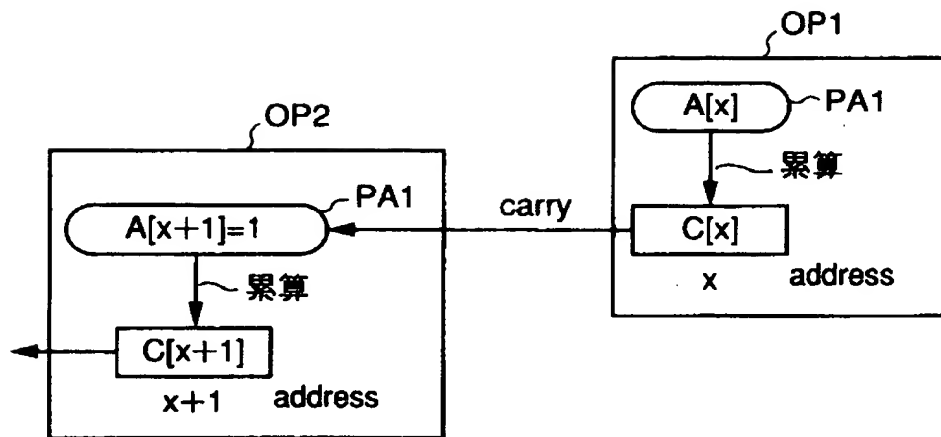
【図 5】



【図 6】



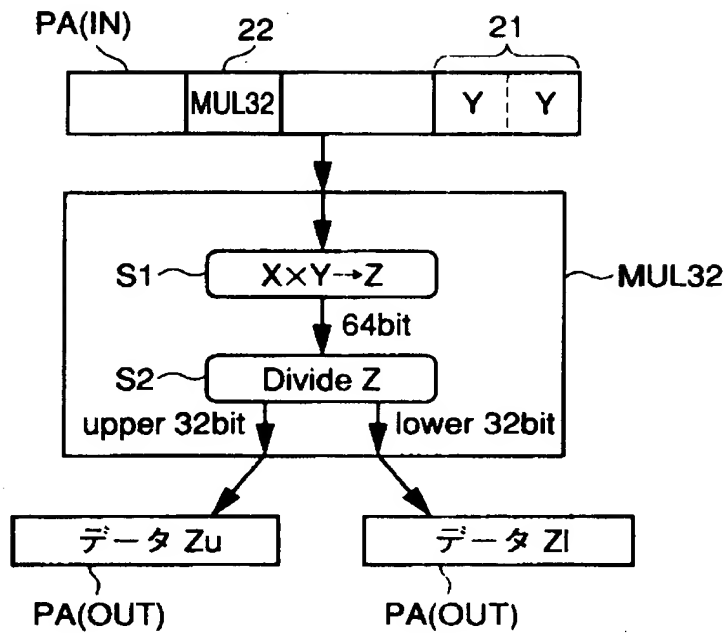
【図 7】



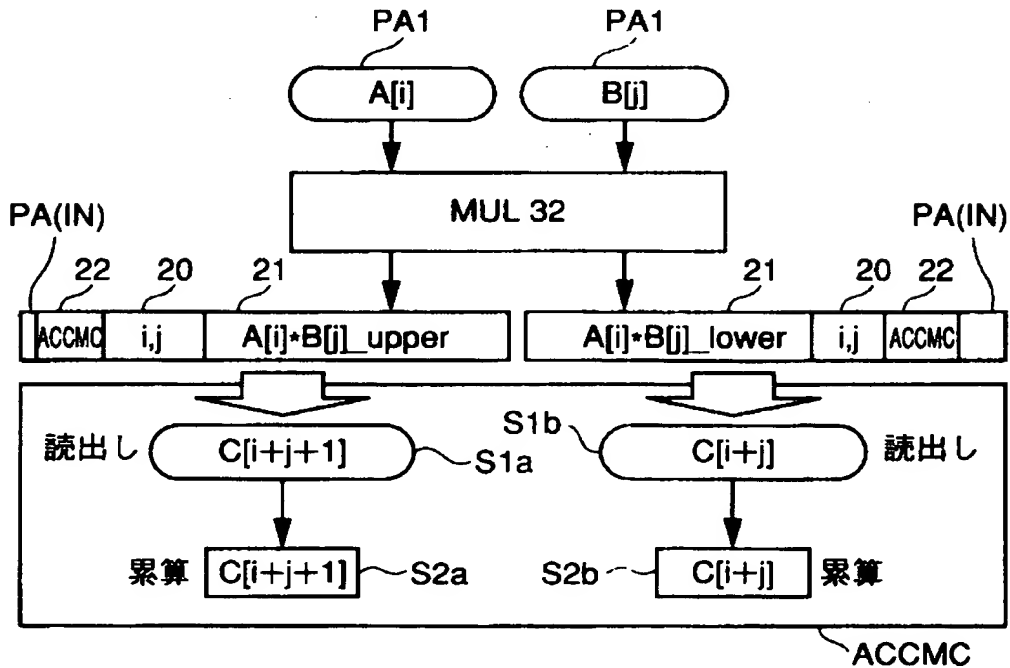
【図 8】

$$\begin{aligned}
 C=A \cdot B = & (A[0] \cdot B[0]) & + (A[0] \cdot B[1]) << (32 \cdot 1) \\
 & + (A[0] \cdot B[2]) << (32 \cdot 2) & + (A[0] \cdot B[3]) << (32 \cdot 3) \\
 & + \dots\dots & + (A[0] \cdot B[31]) << (32 \cdot 31) \\
 & + (A[1] \cdot B[0]) << (32 \cdot 1) & + (A[1] \cdot B[1]) << (32 \cdot 2) \\
 & + (A[1] \cdot B[2]) << (32 \cdot 3) & + (A[1] \cdot B[3]) << (32 \cdot 4) \\
 & + \dots\dots & + (A[1] \cdot B[31]) << (32 \cdot 32) \\
 & + (A[2] \cdot B[0]) << (32 \cdot 2) & + (A[2] \cdot B[1]) << (32 \cdot 3) \\
 & + (A[2] \cdot B[2]) << (32 \cdot 4) & + (A[2] \cdot B[3]) << (32 \cdot 5) \\
 & + \dots\dots & + (A[2] \cdot B[31]) << (32 \cdot 33) \\
 & \dots\dots \\
 & \dots\dots \\
 & + (A[31] \cdot B[0]) << (32 \cdot 31) & + (A[31] \cdot B[1]) << (32 \cdot 32) \\
 & + (A[31] \cdot B[2]) << (32 \cdot 33) & + (A[31] \cdot B[3]) << (32 \cdot 34) \\
 & + \dots\dots & + (A[31] \cdot B[31]) << (32 \cdot 62)
 \end{aligned}
 \quad \left. \vphantom{\begin{aligned} C=A \cdot B = \end{aligned}} \right\} \text{式(1)}$$

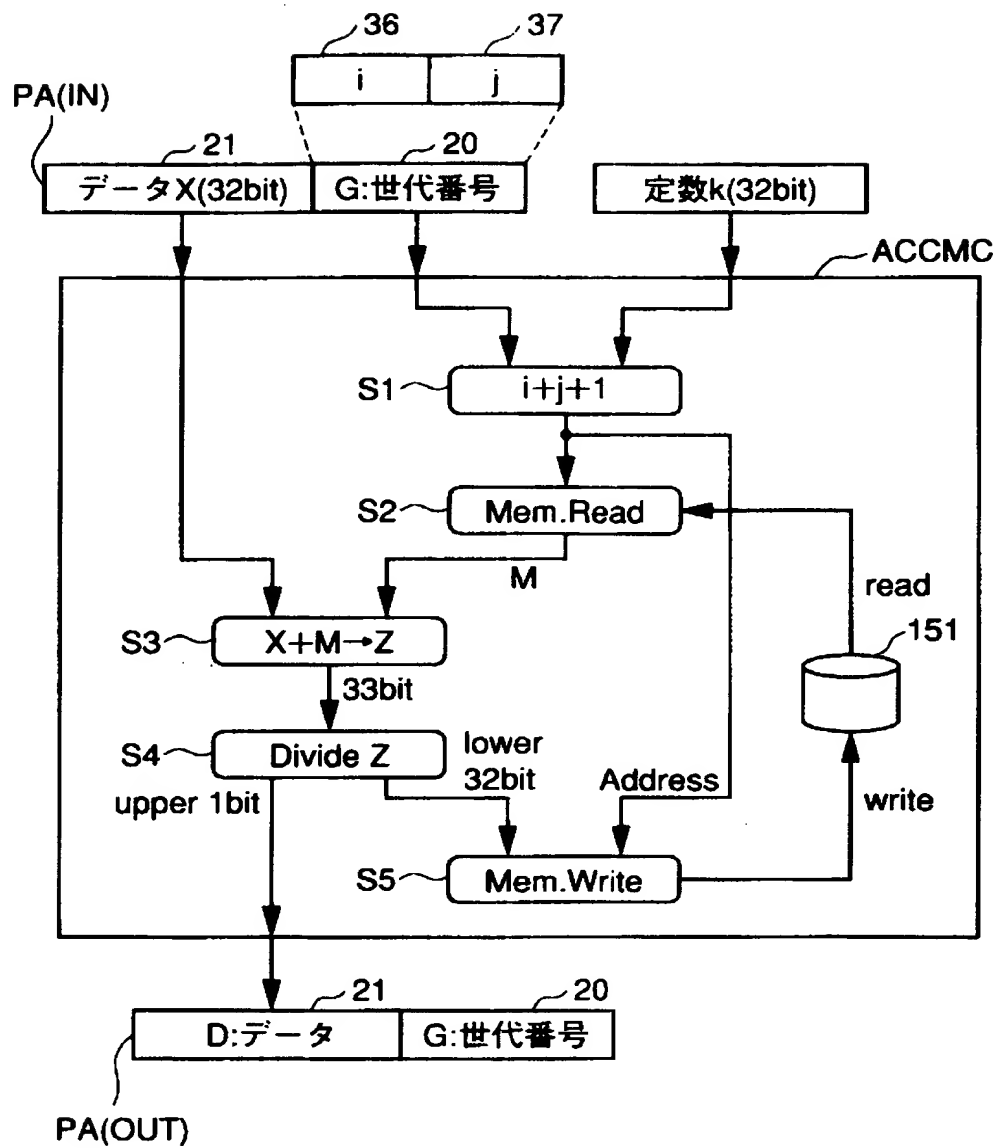
【図 9】



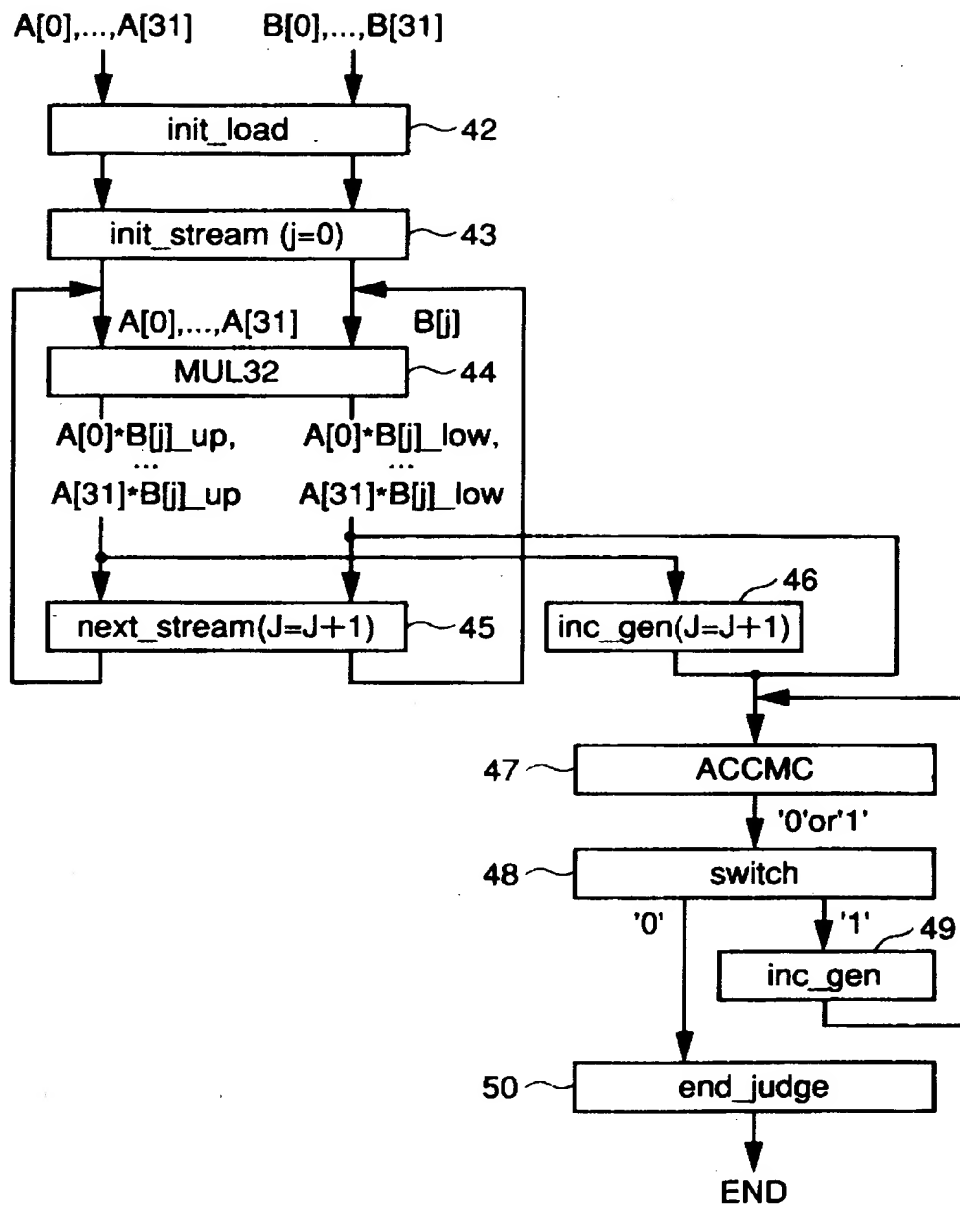
【図 10】



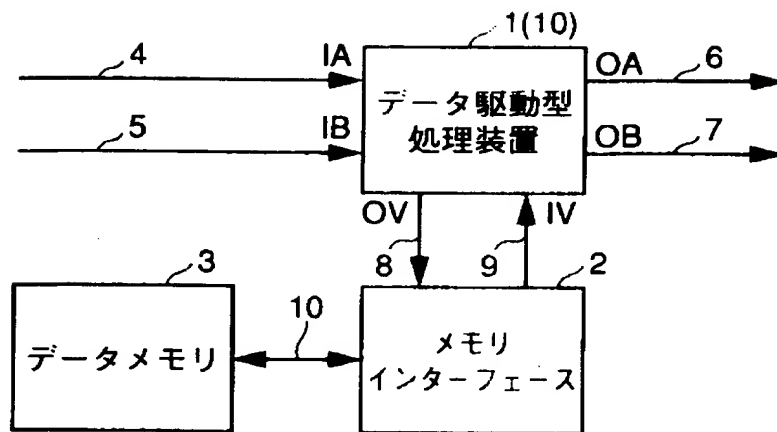
【図 1 1】



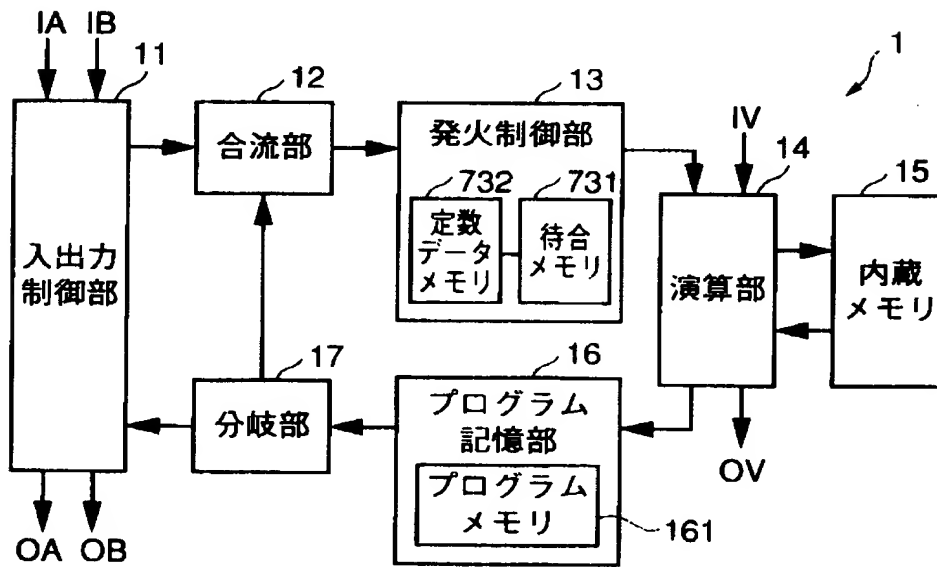
【図 12】



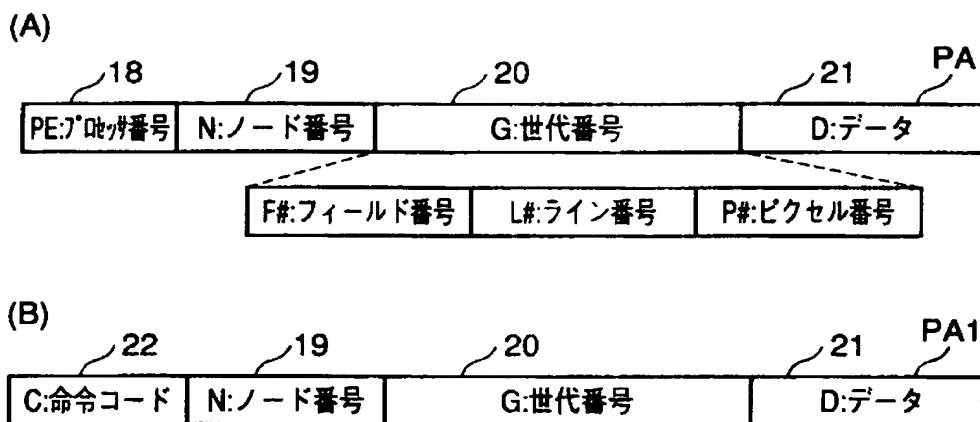
【図 13】



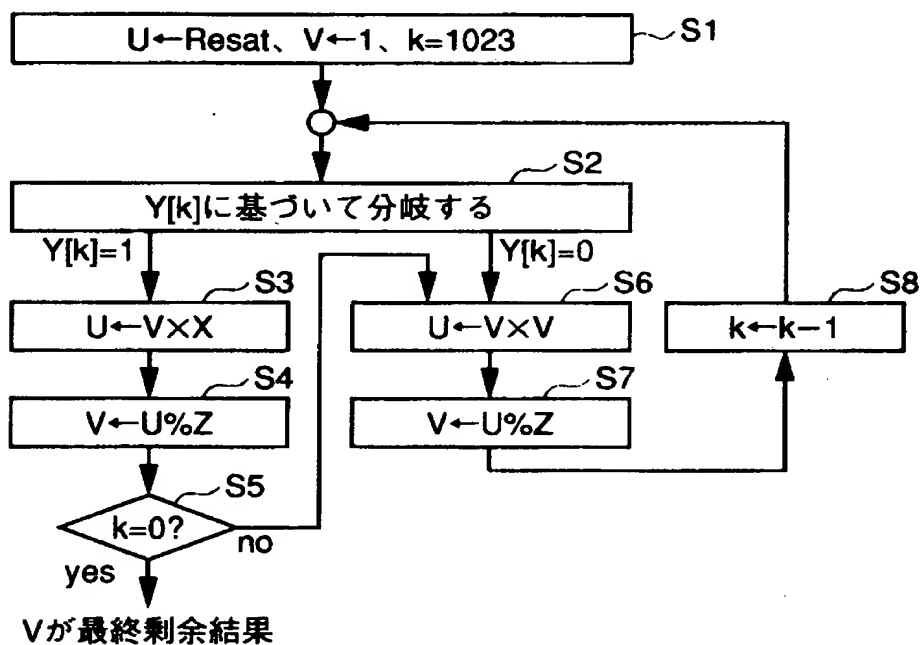
【図 14】



【図 15】



【図 16】



【書類名】 要約書

【要約】

【課題】 データ駆動型処理において多倍精度データを効率よく処理する。

【解決手段】 データ駆動型処理装置において、1024bitの多倍精度データについて所定演算処理が行なわれる際には、多倍精度データは累算メモリ151のメモリワード長に準じて32ビット長単位に分割された複数の単精度データXとして、そしてメモリ151の32ビット長の32個のメモリワードの集合は該多倍精度データとして扱われる。それゆえに、データ駆動型処理装置において多倍精度データ専用の累算器を特別に用意しなくても通常のメモリの領域で多倍精度データの累算器の役割を果たすことができる。また、多倍精度データは32ビット長の単精度データX単位の互いに独立な演算要素に分割することによってデータについての演算をすべて同時並列に実行できるから、データ駆動型処理装置の並列処理能力を最大限に発揮できる。

【選択図】 図11

出 願 人 履 歴 情 報

識別番号 [000005049]

1. 変更年月日 1990年 8月29日

[変更理由] 新規登録

住 所 大阪府大阪市阿倍野区長池町22番22号
氏 名 シャープ株式会社